OASIS – Open architecture for Accessible
Services Integration and Standardization
GRANT AGREEMENT # 215754

Concept Anchoring and Alignment Tool User manual

SubProject No.	**SP1**	SubProject Title	**Open system reference architecture, user interfaces, platform and tools**
Workpackage No.	**WP1.3**	Workpackage Title	**Content Connector and Ontology Management Tools and Interfaces**
Authors (per company, if more than one company provide it together)		**D. Kehagias (CERTH), K. Giannoutakis.**	
Version and Status (F: final; D: draft; RD: revised draft):		**Version 3.0**	
File Name:		**CAAT-usermanual-v3.0.pdf**	
Project start date and duration		**01 January 2008, 48 Months**	

List of Abbreviations

AMI	Ambient Intelligence
API	Application Programming Interface
BPEL	Business Process Execution Language
CAAT	Content Anchoring and Alignment Tool
CCM	Content Connector Module
I/O	Input/Output
IPR	Intellectual Property Rights
JDK	Java Development Kit
RDF	Resource Description Language
SOAP	Simple Object Access Protocol
OR	Ontology Repository
ORATE	Ontology Repository for Assistive Technologies
OWL	Web Ontology Language
UI	User Interface
UML	Unified Modeling Language
WS	Web service
WSDL	Web service description language

Document History

Version	Date	Author(s)
1.0	March 2010	D. Kehagias
1.5	June 2010	D. Kehagias
2.0	September 2010	D. Kehagias, K. Giannoutakis
3.0	October 2011	D. Kehagias, K. Giannoutakis

Table of Contents

LIST OF ABBREVIATIONS **2**

DOCUMENT HISTORY **2**

TABLE OF CONTENTS **3**

LIST OF FIGURES **4**

LIST OF TABLES **6**

1. INTRODUCTION **7**

1.1. ALIGNMENT OF WEB SERVICES 7
1.2. ALIGNMENT OF DEVICES 8

2. INSTALLATION **9**

2.1. REQUIREMENTS 10

3. BASIC CAAT OPERATIONS **11**

3.1. USER AUTHENTICATION 11
3.2. ADD A NEW SERVICE 13
3.2.1. An example of use *17*
3.3. MANUAL SERVICE INVOCATION 18
3.3.1. Special Cases *20*
3.4. EDIT WEB SERVICE METADATA 23
3.5. EDIT SERVICES 24
3.6. CAAT BACKGROUND PROCESSES 25

4. CREATING AND TRAINING NEW CLASSIFICATION MODELS **27**

4.1. CREATE A NEW APPLICATION DOMAIN 27
4.2. SERVICE ONTOLOGY 30
4.2.1. Manual Creation of a new Domain *32*
4.3. CREATE A NEW "IDEAL" OPERATION 32
4.3.1. Create a new ideal operation in CAAT *32*
4.3.2. Add a new ideal operation in the Service Ontology *33*
4.3.3. Add new concept classes that describe the I/O of the operation *35*
4.3.4. Training *35*

5. CREATING BUSINESS RULES **37**

5.1. SUPPORTED BUSINESS RULES 37
5.2. DEFINING BUSINESS RULES IN CAAT 38

6. BUSINESS PROCESSES-WEB SERVICE CASCADING **41**

6.1. INTRODUCTION 41
6.2. LINKING IDEAL OPERATIONS 43
6.3. HOW TO CREATE AND EDIT A BUSINESS PROCESS 45
6.4. BUSINESS PROCESSES AND BUSINESS RULES 50
6.5. EXECUTION OF A BUSINESS PROCESS 51
6.6. IMPORTING EXTERNAL BPEL PROCESSES 54

7. REFERENCES **56**

List of Figures

Figure 1. A typical WSDL file. In this example we show the *GetForecastByZip* operation that includes the *GetForecastByZipSoapIn* input and *GetForecastByZipSoapOut* output parameter. The details of these parameters (e.g. their data types) are defined under the *message* element. 8
Figure 2. File ccm.properties. In this file the user may appropriately define values for runtime parameters, such as the location of the CCM database (service registry), the location of various auxiliary folders that are created dynamically and a set of other parameters concerning the functionality of the tool. 9
Figure 3. Initial CAAT window that prompts the user to enter a username and a password. 11
Figure 4. CAAT user registration window 12
Figure 5. CAAT home screen 12
Figure 6: A user enters the URL of a web service 13
Figure 7. A warning message that appears when the user tries to align an already aligned service. 14
Figure 8: 1st Level Classification results 14
Figure 9: Manual selection of the application domain 14
Figure 10: Manual selection of the application domain 14
Figure 11: Selection of the classification algorithm 15
Figure 12. 2nd and 3rd Level Classification results. In case one or more of the aligned operations are matched incorrectly the user has the option to manually correct the ontology operations that match with the real ones by using a drop-down list of all available “ideal” operations. 15
Figure 13: By clicking on any real operation parameter the aligned parameter of the corresponding ideal operation is highlighted. 16
Figure 14. CAAT drag-and-drop functionality allows users to manually correct the result of the automatic alignment process. 17
Figure 15: Warning message when not all inputs or outputs are aligned during saving. 18
Figure 16: WSDL operation list with one operation aligned. 18
Figure 17: Web Service Invocation panel 19
Figure 18: The results that are returned after the invocation of the real web service operation are shown both in the form of a SOAP message or by their content whenever this is possible. 20
Figure 19: Example of an array element `Stop`. User is prompted to add elements to array. 21
Figure 20: Inputs of the operation when user adds additional elements to array. 21
Figure 21: The definition of an element that has *enumeration* values. 22
Figure 22: Invocator identifies enumeration values, and allows the user to select specific values from the allowed ones. 22
Figure 23: The list of allowed values defined in the Enumeration. 23
Figure 24. By the third option the user may insert metadata to describe web service properties. These are used for personalisation purposes. 24
Figure 25: Various forms for inserting metadata parameters. 25
Figure 26: The aligned operations panel. 26
Figure 27: The service provider has the option to invoke his aligned operations, or to delete his aligned web services. 26

Figure 28. By pressing the Change Domain button the user may select to create a new domain.....27
Figure 29. New Domain Name and description option. 27
Figure 30. As soon as the ontology is uploaded on the ORATE repository, a confirmation message shows up. This means that the ontology was been successfully uploaded. 28
Figure 31: The user presses the “upload training data for domain” button in order to provide the web services to be used as training data for training the new domain.....29
Figure 32: The user is prompted to upload wsdl files to the training data repository .29
Figure 33: When creating a new domain, equal number of ideal operations to the wsdl operations are created, but no alignment can be saved before the domain approval by the administrator. 30
Figure 34. Training a first-level classification model. 31
Figure 35. (a) Snapshot of the Web-service ontology in Protégé, (b) some of the most important object properties in Manchester notation 31
Figure 36. Add new domain in the Service Ontology using the Protégé ontology authoring tool. 32
Figure 37. Create a new ideal operation using the CAAT user interface. 33
Figure 38. Creating a new ideal operation in Protégé.....34
Figure 39. Properties that contain information about I/O of operations.....35
Figure 40. Create new classification model panel. The user may select the parameters of the data preparation and training processes that lead to an appropriate classifier...36
Figure 41: Service Providers may define business rules to his operation at third classification level or at the metadata frame. 38
Figure 42: The business rules editor. 39
Figure 43: Drag and drop functionality in order to add WSDL files allowed for cascading. 40
Figure 44. The result of cascading mechanism between two operations *x*, *y*. The aligned pairs (i_1-o_3) and (i_3-o_4) result in a unified cascaded operation with two inputs and two outputs. 41
Figure 45: The cascading mechanism using ideal operations matching. The two inputs of the initial operation are finally cascaded with outputs of two already aligned operations in CCM through ideal operation I/O matching. Finally, the inputs needed for “Operation 1” to be invoked is only “in6”. 42
Figure 46: “Link ideal operations” main window.....43
Figure 47: An error message appears when the user tries to align two input or two output parameters. 43
Figure 48: Confirmation window for linking two ideal operation parameters. 44
Figure 49: Link ideal operations window. 44
Figure 50: Confirmation window for linking two ideal operation parameters. 44
Figure 51: Business process editor. Right toolbox includes the basic operations that the editor supports. In the main area a business process may be designed using basic workflow components.....45
Figure 52: By right clicking on the business process canvas area a list of available options appears.....46
Figure 53: The different types of nodes that may appear in a business process. 46
Figure 54: When a new Invoke node is added to the business process a window appears that prompts the user to select, among the aligned operations, the one that should participate in the business process. 47

Figure 55: When a new *Invoke* node is added to the business process a prompt window appears. 48
Figure 56: With drag and drop functionality the user can align an input (or an output) shown on the left to an output (or an input, respectively) shown on the right....48
Figure 57: With drag and drop functionality the user can specify an *if* statement (create a logical expression)....48
Figure 58: The user may select which output objects will be returned via a *Return* node....49
Figure 59: Actions toolbar 49
Figure 60: Load business process window 50
Figure 61: Business rules toolbar....50
Figure 62: In the "Set business rules" window the pricing policy associated with the use of the business process is determined.....51
Figure 63: View total cost window.....51
Figure 64: Manual assignment of values to input parameters. 52
Figure 65: The business process execution is visible on the canvas....53
Figure 66: Simulation properties window....53
Figure 67: Business process output window....53
Figure 68: Available business processes of operation "getTaxiWithCoordinates".....54
Figure 69: A part of a BPEL process file.....55
Figure 70: Graphical representation of a BPEL process, in the Netbeans® BPEL designer. 55

List of Tables

Table 1. Meta-data parameters....23
Table 2: Specific Business Rules for web services.....38

1. Introduction

The purpose of Concept Anchoring and Alignment Tool (CAAT) is to align the functionality of the provided services and /or devices with the ontologies stored in the ontology repository (i.e. ORATE). The concepts of the same or different application areas, after being aligned with other ontological concepts, are able to anchor in the hyper ontology, thus being ready to be used seamlessly through the tool. Modifications to the aligned concepts are feasible using a user-friendly functionality that hides the complexity of the conceptualised connections from the user.

1.1. Alignment of Web services

The tool provides a fast and easy way to classify SOAP web services based on their WSDL description files. As opposed to the most common available service classification techniques, which deal with the classification of WSDL documents into different application domains, the technique developed for CAAT's purposes focuses on the classification of all structural elements of a WS. i.e. it structures operations and their input and output parameters into different classes that describe operations of the same application domain, and their parameters respectively. A typical view of a WSDL file is illustrated in Figure 1. Appropriate XML structures are used to annotate the various WSDL structural elements. Amongst these, a WSDL file uses the *operation* XML element in order to annotate WS operations, while the *input*, *output* elements are used for the definition of the corresponding I/O operations. A WSDL file also includes other elements in order to specify the invocation details of its operations.

Thus, any WSDL description represents a WS with the following hierarchy:

- The top level of the hierarchy is represented by the WS that is structured as a collection of operations and potential data type definitions.
- The WS operations that represent methods or functions that are invoked by the WS clients belong to the next hierarchical level.
- The bottom-most level includes the input and output parameters of the WS operations.

With respect to the aforementioned analysis of the basic hierarchical levels that represent the internal WSDL structure, we introduce a semantic classification schema composed of the following three levels.

- *1st classification level*. It concerns classification of a service into a particular application domain.
- *2nd classification level*. It encounters matching of WS operations with the so-called "ideal" operations" that are defined in some of the ontologies that constitute ORATE.
- *3rd classification level*. On this level, WS operation inputs and outputs are matched with ontologically defined input and output parameters.

CAAT has been implemented as a standalone application in Java. Its layout is based on a wizard-based user friendly approach to facilitate users on the execution of the steps required for the completion of the alignment process.

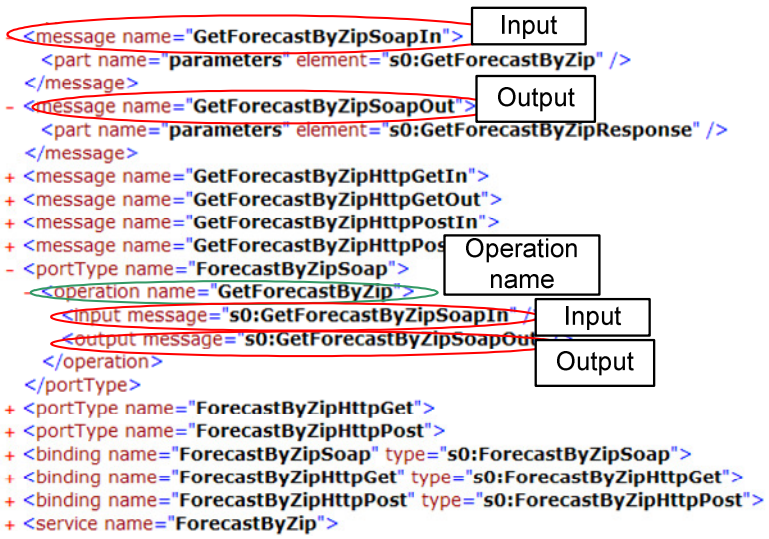


Figure 1. A typical WSDL file. In this example we show the *GetForecastByZip* operation that includes the *GetForecastByZipSoapIn* input and *GetForecastByZipSoapOut* output parameter. The details of these parameters (e.g. their data types) are defined under the *message* element.

In the ontology appropriate associations between ontological concepts are defined in the form of object properties to sufficiently annotate real WS and their structural elements. For example, the *hasInput*, object property is defined in order to associate a WS operation with its input parameters.

The classification of a WS in a given domain, which is undertaken on the first layer, implies that all operations that are included in the WSDL document belong to the same domain as well. In the second classification layer, any WSDL document that represents a WS is analysed into its operations. The purpose of automatic classification of operations is to allow automated annotation of operations apart from WS classification and thus facilitate more automated ways of service invocation. Architecture – Positioning with respect to the other components.

The CAAT has been implemented as a standalone application in Java. However it communicates with the OR through the CCM that operates as a middleware. Also CAAT communicates to the service registry that has been implemented as a rational database system, which is part of the CCM. The communication between the CAAT and the CCM is performed via software interfaces that are shared between the middleware and the tool.

1.2. Alignment of devices

CAAT provides the capability to align ontological descriptions of devices against semantically corresponding concepts in the hyperontology, in a way similar to the one applied for the alignment of web services. The alignment process is undertaken by the same algorithms used for web service alignment. In order to enable alignment of devices, each device to be aligned should be linked to a web service interface that is used in order to control the device. Typical operations that a "device" web service should support include *readValue*, *activateDevice*, *shutDownDevice*, and so on. Thus, CAAT treats devices as web services. Any device web service belongs to the same domain, which is called *"Device"*. Therefore, the result of aligning devices will be a set of ontology mappings between a newly incoming device (described as a web service) and the corresponding description (in the form of an "ideal" operation) defined in the hyperontology in the *"Device"* domain.

2. Installation

Step 1 Unzip file CAAT.zip in a folder (e.g. D:/CAAT).

Step 2 (optional) CAAT requires the OASIS Content Connector Module to be up and running. We assume that the CCM is located at a specific central point whose IP address should be specified in the CAAT configuration file *ccm.properties* located in the CAAT installation directory. A snapshot of this file is illustrated in Figure 2. In this file the user may appropriately define values for runtime parameters, such as the location of the CCM service registry, the location of various auxiliary folders that are created dynamically (such as the ones used by training and create new classification model processes), as well as a set of other parameters concerning the functionality of the tool. If you leave the default values the CAAT will be connected to the default CCM (with IP address: 160.40.50.57)

Step 3 In order to run CAAT execute file **run.bat** in the CAAT installation directory.

```
CCM_DomainSet=domains/
CCM_secondLevel_Feature_Occurence_Expression=Frequency
CCM_LocalWSDLFileServiceUrl=http\://160.40.50.57\:8080/CCM_LocalWSDL/CCMWS_LocalWSDLService?WSDL
CCM_WSDLWORDSParsingLevel_OperationsIONames=1
CCM_StemWords=1
CCM_OntologyRepository=ontology/
CCM_secondLevel_WSDLWORDSParsingLevel_OperationsIONames=1
CCM_TestSet=tests/
CCM_UseNegativeDictionary=1
CCM_AlgorithmLevelThree=0
CCM_secondLevel_StemWords=1
CCM_secondLevel_WSDLWORDSParsingLevel_OperationsDocumentation=1
CCM_WSDLWORDSParsingLevel_OperationsDocumentation=1
CCM_DatabaseName=oasisdb
CCM_AlgorithmLevelOne=0
CCM_secondLevel_MinimumWordLength=2
CCM_MinimumWordLength=2
CCM_Feature_Occurence_Expression=Frequency
CCM_TrainSet=training/
CCM_ServiceProviderProfileWSDLUrl=http\://160.40.60.230\:8080/axis2/services/ProviderProfileWS?wsdl
CCM_DatabaseIP=160.40.50.57
CCM_SingularizeWords=1
CCM_secondLevel_WSDLWORDSParsingLevel_OperationsNames=1
CCM_WordNetDirectory=Wordnet/2.0/dict/
CCM_User=oasisuser
CCM_ExperimentResults=experimentResults/
CCM_AlgorithmLevelTwo=0
CCM_secondLevel_UseNegativeDictionary=1
CCM_HOME=./
CCM_secondLevel_SingularizeWords=1
CCM_OntologyImportsRepository=imports/
CCM_Pswd=oasis123
CCM_OrateUrl=http\://ontologies.informatik.uni-bremen.de\:8080/bioportal/ontologies
CCM_UseOrate=1
CCM_secondLevel_NegativeDirectory=Wordnet/2.1/dict/
CCM_WSDLWORDSParsingLevel_OperationsNames=1
CCM_NegativeDirectory=Wordnet/2.1/dict/
CCM_TrainedModelDir=models/
CCM_TrainedModelDir2ndLevel=models2ndLevel/
```

Figure 2. File ccm.properties. In this file the user may appropriately define values for runtime parameters, such as the location of the CCM database (service registry), the location of various auxiliary folders that are created dynamically and a set of other parameters concerning the functionality of the tool.

2.1. Requirements

CAAT executable has some requirements in order to run properly. In order to communicate with ORATE, CCM and service registry, the terminal that runs CAAT should have access to the ports 3306, 4848 and 8080.

Additionally, it should be mentioned that CAAT executable encounters some GUI performance issues when running under JDK version 6.20. Therefore, JDK version 6.21 is recommended.

3. Basic CAAT operations

The CAAT is a tool intended for the following users:

- service providers who are willing to register a new service into the OASIS platform,
- device developers who want to align the functionality of their devices into the OASIS ontology.

By registering a new service through CAAT, the service becomes available to the OASIS platform, thus it is capable to be invoked by any end-user application. By adding a new device, the device is mapped in the hyperontology and becomes visible in the ontology repository. Other users could also make use of CAAT (such as researchers and practitioners) in order to test the alignment processes between services and for research and improvements of the alignment process. When running CAAT for the first time, the tool asks for authentication.

3.1. User authentication

CAAT supports user registration in order to assure that only authenticated users enter the system and store user information. User registration is required because only authenticated users may register a new service or device into the OASIS platform or remove assets from it. In the initial CAAT window the tool prompts the users to enter using their credentials (Figure 3). At the same time CAAT downloads automatically the latest ontology from the ORATE ontology repository.

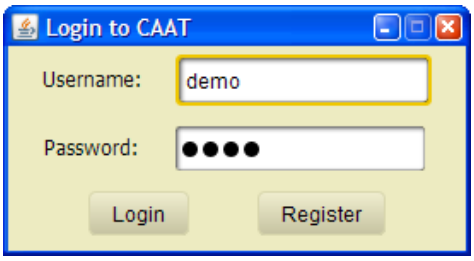


Figure 3. Initial CAAT window that prompts the user to enter a username and a password.

If the user does not have any account he/she may create one by pressing the Register button and entering the required data in the form that appears (Figure 4). After successful login, users may choose one of the following basic functionalities performed by the tool, as shown in Figure 5:

(1) integrate services into the platform;
(2) manually invoke one registered service,
(3) edit information (meta data) about the aligned WSs,
(4) edit already aligned services,
(5) change personal information,
(6) link “ideal” operations (i.e. operations defined in the ontology),
(7) create new business processes,
(8) download the latest ontology,
(9) log out.

The users may edit their profile at any time by pressing the “Edit Profile” button on the CAAT’s home screen (Figure 5). By pressing the “Log out” button the user end its CAAT session and the initial login window appears again. Each one of the remaining functionalities is described in the next Sections of this manual.

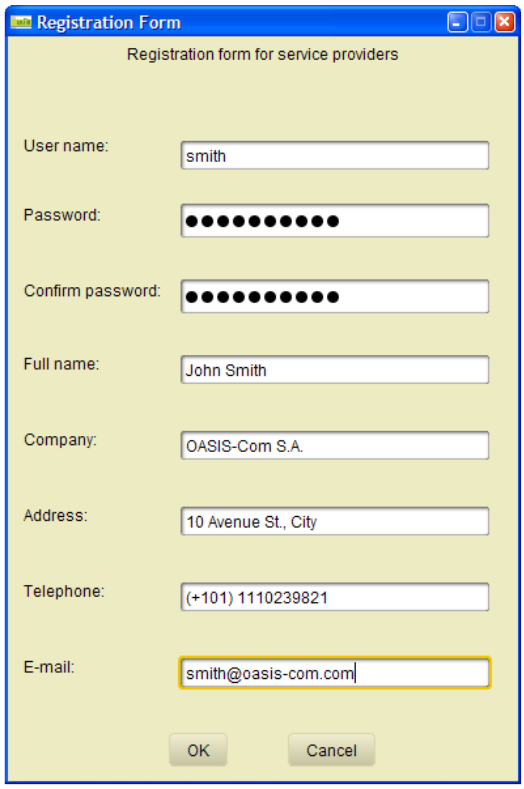


Figure 4. CAAT user registration window

Figure 5. CAAT home screen

3.2. *Add a new Service*

The main functionality supported by CAAT is to allow service providers add a new service into the OASIS service registry. The new service will be semantically aligned according to the Service ontology, which is imported from the ontology repository. A service provider who wants to add a new service should select the option "Integrate your service into the OASIS framework" in the initial CAAT option window (Figure 5). The only required information for a service to be aligned is its URL. In the example shown in Figure 6 the user enters the following URL:

http://160.40.50.81/WebServices/Service.asmx?WSDL

that corresponds to a valid web service in WSDL format. Currently the tool supports alignment of SOAP services only. Their semantic classification that results in the final alignment of their structure elements is based on information included in the WSDL description file.

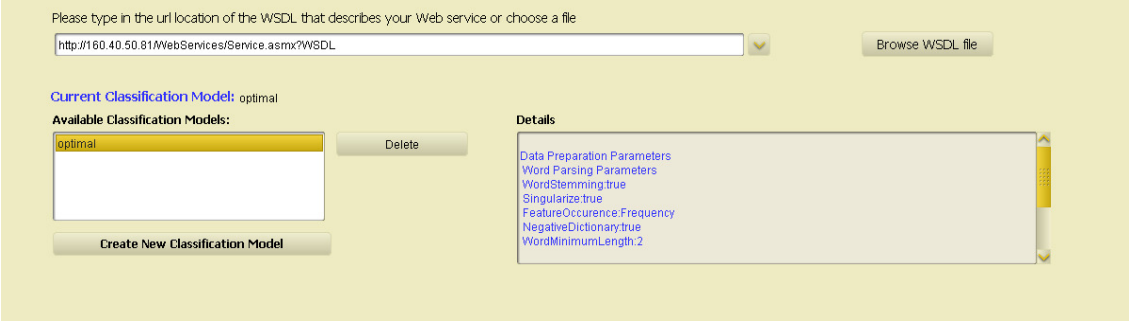


Figure 6: A user enters the URL of a web service

In the panel shown in Figure 6 the current classification model is shown. This model has been generated in a previous training session and it is necessary for the execution of the classification process that will be performed in the next step of the alignment procedure. At this point the user has either the option to continue in the next step, i.e. to launch the alignment procedure, or to create a new classification model by pressing the corresponding button. This part should be normally transparent to the users but it is included as a new feature in the tool in the last version of the tool in order to allow experimentation with various methods and techniques. This functionality exists for research purposes, (e.g. to draw useful conclusions about which algorithm or combination of algorithms and corresponding parameters lead to an optimised solution regarding the concept matching accuracy, according to the three aforementioned classification layers). The details of the new model creation and training are described in chapter 4.

It should be noted that if the selected classification model does not correspond to the available domains (e.g. the model was created in the past with less domains compared to the ones that now exist), then a warning message will appear.

By pressing the *Next* button the tool checks if the current web service is already aligned (by another web service provider) and if this holds, an error message appears preventing the user to align this web service (Figure 7). After this check and if the web service is not aligned by any other service provider, the classification process runs and the results appear after a few seconds on the window illustrated in Figure 8. In order for a web service to be aligned, it should be up and running. In case as service is not operational (i.e., the service is offline) an error message appears, such as the one shown in Figure 9.

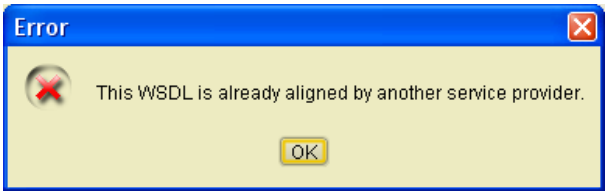


Figure 7. A warning message that appears when the user tries to align an already aligned service.

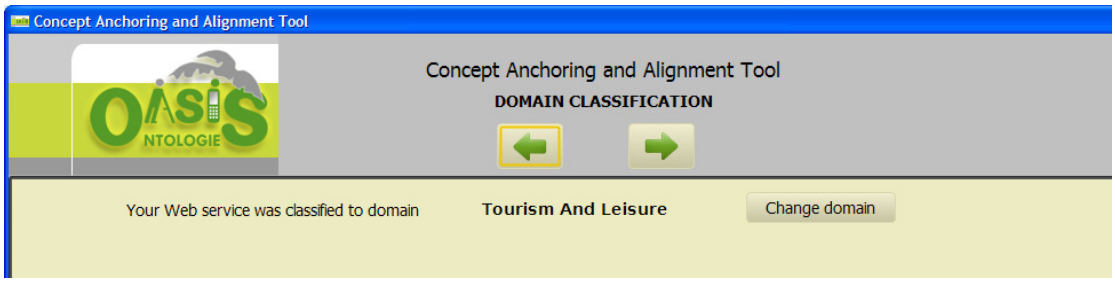

Concept Anchoring and Alignment Tool
Concept Anchoring and Alignment Tool
DOMAIN CLASSIFICATION
Your Web service was classified to domain
Tourism And Leisure
Change domain

Figure 8: 1st Level Classification results

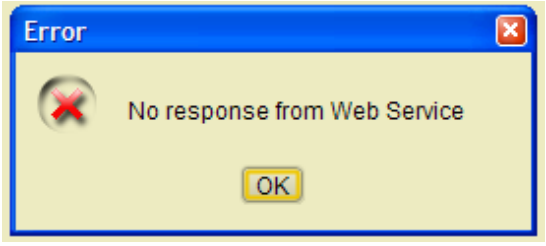


Figure 9: Manual selection of the application domain

As it is shown on this window, the CAAT provides the option to **manually change the domain** to which a web service is classified in the first step (*1st level classification*). By clicking on the "Change domain" button, the user may select an alternative domain from the drop-down list that appears in Figure 10. If an operation of the current WSDL file is already aligned to another domain, a warning message appears on the status bar, while the *Next* button is deactivated, prohibiting the user to align the operation to another domain. The user has also the option to manually create a new domain, in case that none of the existing domains adequately describes the current WS. The details of this process are outlined in Section 4.1.

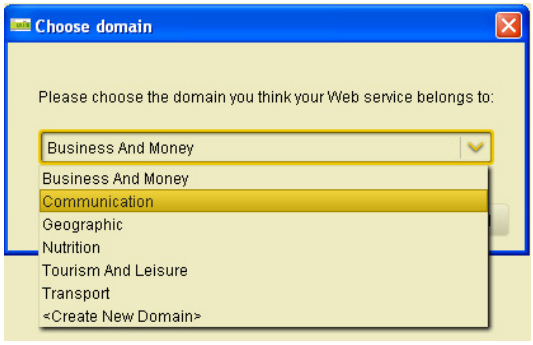


Figure 10: Manual selection of the application domain

After completing the first step, the user may proceed to the next one that deals with the alignment of the WS operations into their ontological counterparts. The first

action required by the user in the next wizard window is to select the classification algorithm to be used for the classification of the WS operations. The user selects one out of the three options shown in Figure 11.

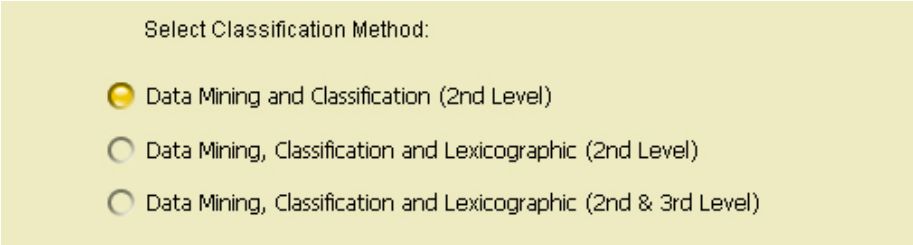

Select Classification Method:
Data Mining and Classification (2nd Level)
Data Mining, Classification and Lexicographic (2nd Level)
Data Mining, Classification and Lexicographic (2nd & 3rd Level)

Figure 11: Selection of the classification algorithm

The selection of the appropriate classification algorithm leads to the next step in the CAAT wizard that launches the classification of each one of the WS operations. This is also known as 2nd and 3rd level alignment process. An example is illustrated in Figure 12.

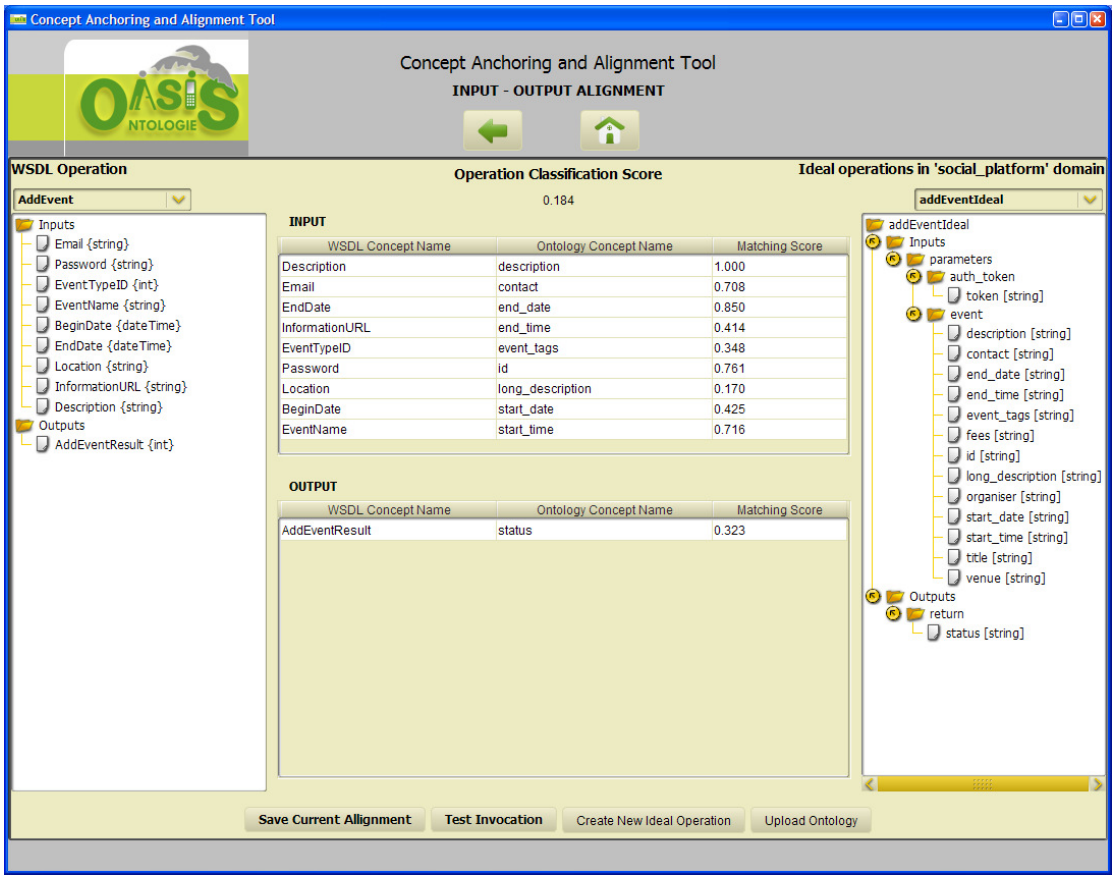


Figure 12. 2nd and 3rd Level Classification results. In case one or more of the aligned operations are matched incorrectly the user has the option to manually correct the ontology operations that match with the real ones by using a drop-down list of all available "ideal" operations.

In the table depicted in Figure 12 the real WS operations are listed in the drop-down list labelled "WSDL Operation", while the matched operations in the ontology are

shown under the label "Ideal operations in *domain*", where *domain* is the name of the domain in which the web service has been classified. By clicking on any real operation parameter, the corresponding aligned parameter that belongs to the ideal operation is highlighted (Figure 13). In the column labelled "Operation classification score" a normalised measure of the accuracy of the matching process is listed for each one of the matched operations. Finally, the user has the option to accept or reject a given alignment proposed by the tool automatically. In the latter case the option of manual correction of the resulted matching is still available. If the alignment that the system recommends is not valid, the user may change it by using **drag-and-drop** functionality operations (see e.g. Figure 14).

After the completion of this stage the user is able to select and save each one of the desired aligned operations, by pressing the "Save current alignment". When an alignment is saved the tool creates for the selected WSDL operation the corresponding entries in the aligned services registry. Additionally, it creates and uploads to OR the corresponding ontology file that describes the alignment between the WSDL real operation and the corresponding ideal operation.

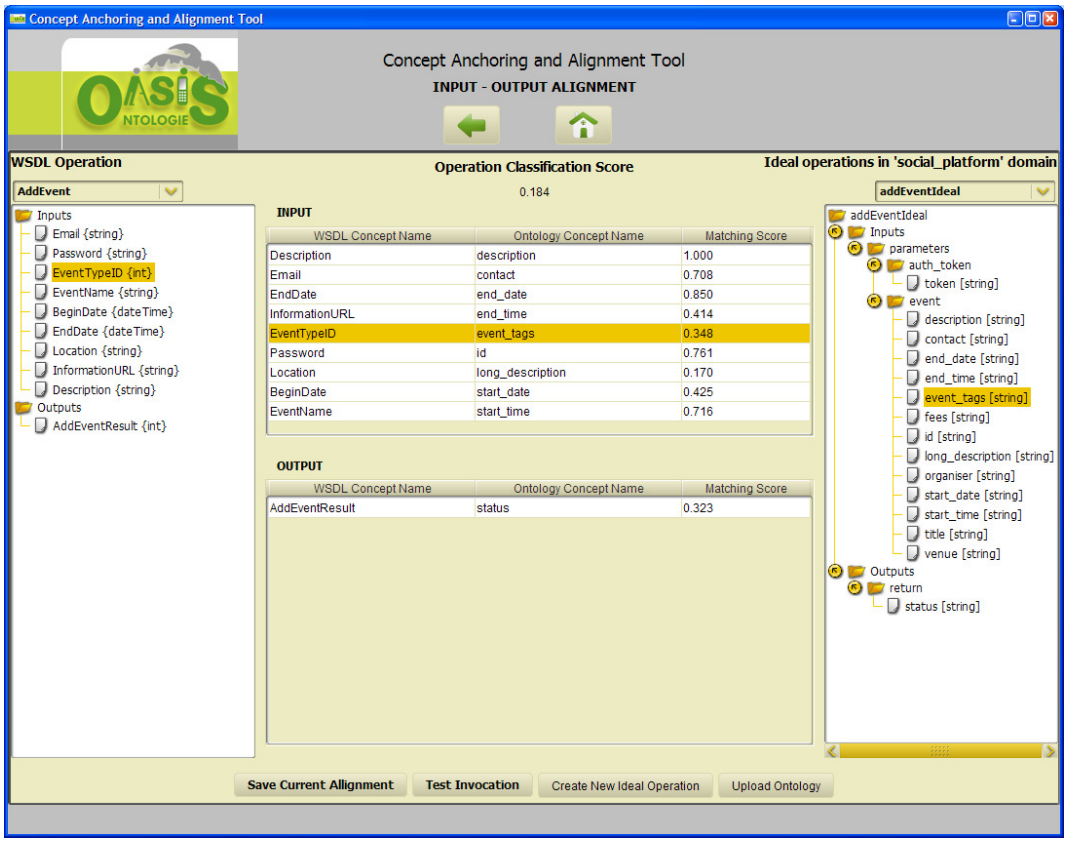


Figure 13: By clicking on any real operation parameter the aligned parameter of the corresponding ideal operation is highlighted.

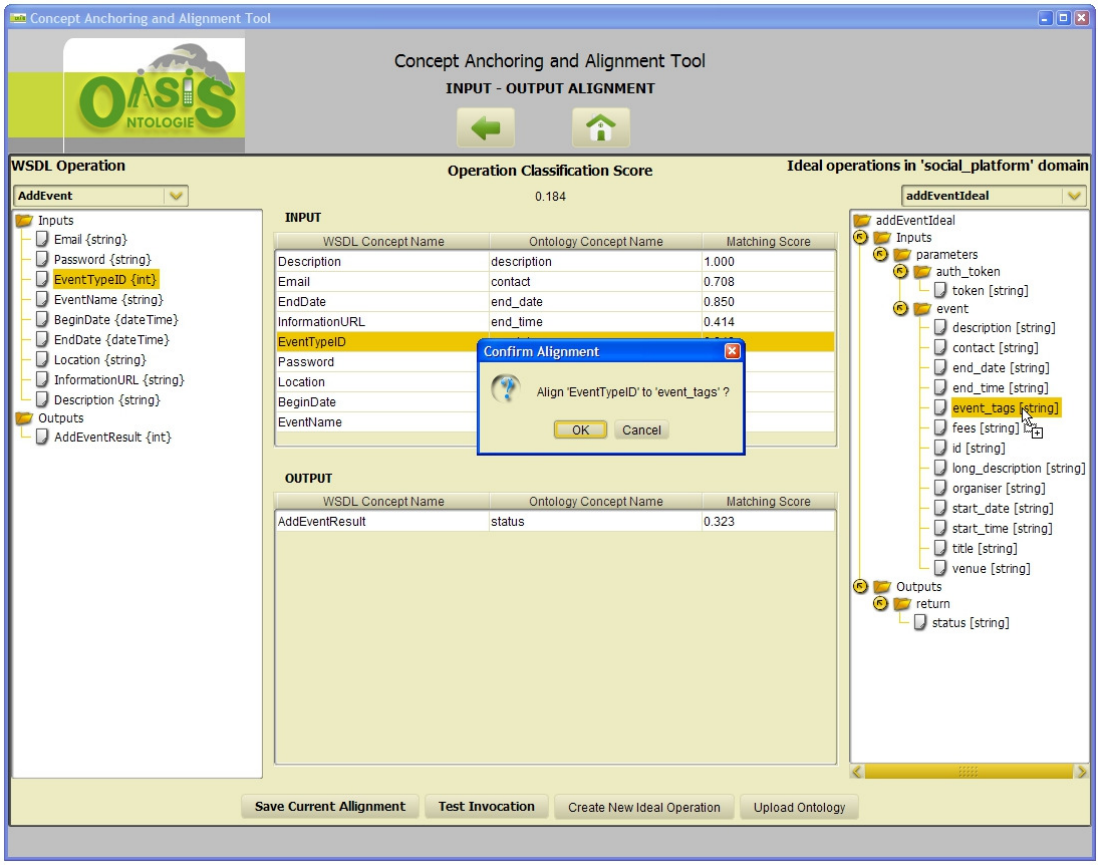


Figure 14. CAAT drag-and-drop functionality allows users to manually correct the result of the automatic alignment process.

3.2.1. An example of use

The following example illustrates the process followed by a service provider in order to enter a new service into the system.

Let us assume that a service provider launches the application and logins into the system (Figure 3). In the window that appears use selects “Integrate your service into the OASIS framework” option (Figure 4) and presses the *Next* button. The user inserts the url of his working web service that corresponds to a valid WSDL description (Figure 6) and presses the ***Next*** button. The tool classifies the web service into the “Tourism and Leisure” domain (Figure 8). He presses the ***Next*** button in order to proceed with the next window.

The Data Mining selection method is selected by default (Figure 10). The user presses the ***Next*** button and in the next window (Figure 13) the various alignments suggested by the tool for the included web service operations are shown. The two drop-down lists allow the selection of a particular ideal and real operation respectively. The right drop-down list containing all ideal operations from the classified domain is ranked as follows:

1) If the operation is already aligned, the aligned ideal operation will appear first on the list, and
2) the remaining ideal operations are displayed in descending order, according to the classification score of the second classification level.

The user selects one real operation (e.g. *GetTotalSocialEvents*) that is aligned to the ideal operation called *getSocialEventList*. Among the output parameters of the real operation the one called *month* is incorrectly matched to the *date* instead of the parameter *month*, which is the correct one. In order to manually correct this mismatch the user uses the ***drag-and-drop*** functionality that the tool supports dragging the item called *month* from the left list and dropping it on the country item of the right list, as shown in Figure 14. A dialog window then appears to ask for user's confirmation on the previous action.

Finally, by pressing the "Save current alignment" button, the alignment of the particular operation is saved into service registry, as well as in the form of an ontology file that is uploaded as a mapping on the ontology repository. If not all inputs and outputs are aligned with concepts in the ontology, then a warning message appears, as shown in Figure 15. The already aligned operations appear in bold font in the left drop-down list (Figure 16).

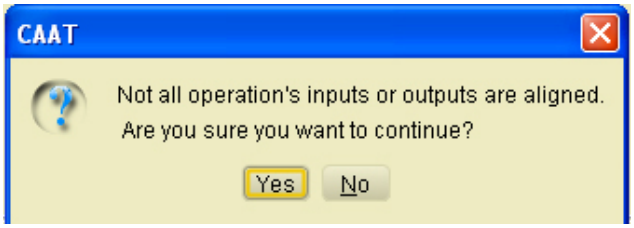


Figure 15: Warning message when not all inputs or outputs are aligned during saving.

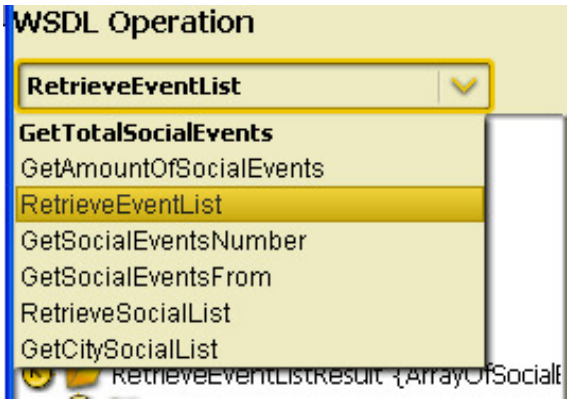


Figure 16: WSDL operation list with one operation aligned.

3.3. Manual Service Invocation

The user has the option to check out the invocation of any operation by manually inserting values to their input parameters. This option is included in CAAT for testing purposes and in order to make sure that each one of the aligned operations is able to be invoked and returns the appropriate content/value. This option is available to the user either on the initial option window shown in Figure 3 for aligned operations, or at the 3rd level classification process through the "Test invocation" button. The invocation panel is shown in Figure 17. By using the invocation form from the initial window, the user is able to view also the aligned concepts of the ontology.

On the WS invocation panel the user may select an "ideal" operation that has been aligned with real WS operations. For the selected "ideal" operation a list of possible

inputs is shown. Among those inputs the user may find the ones that have been matched/aligned to the input parameters of the aligned operations after the 3rd level classification process. The WS invocation panel allows users to insert values to those parameters that correspond to real WS operations. In the example shown in Figure 17 the user inserts the value 1980 into the "year" input parameter. The user can switch the view to the real operation by selecting "Wsdl operation" at the drop-down list. When the minimum required number of input parameters has been assigned to real values there is at least one real WS operation that can be invoked. The invocation of the real WS operation is performed by pressing the "Invoke" button. The results that are derived from the real WS after the invocation of its selected operation are illustrated in Figure 18.

These are shown both in the form of a SOAP message (SOAP request and SOAP response) or in terms of their content, whenever this is possible (e.g. when an output can be expressed as a known primitive data type such as integer, string, etc.). In the example shown in Figure 18 after the WS invocation the value of the output parameter called "GetTotalSocialEventsResult" is the integer number 4.

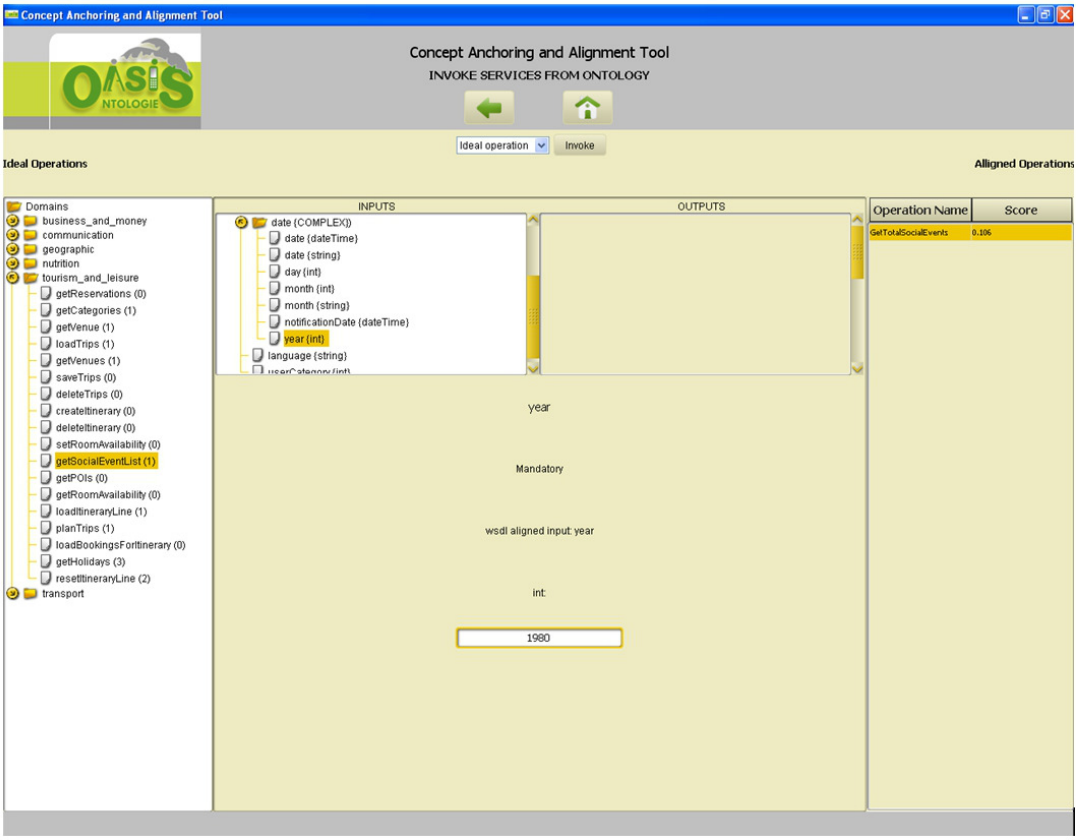


Figure 17: Web Service Invocation panel

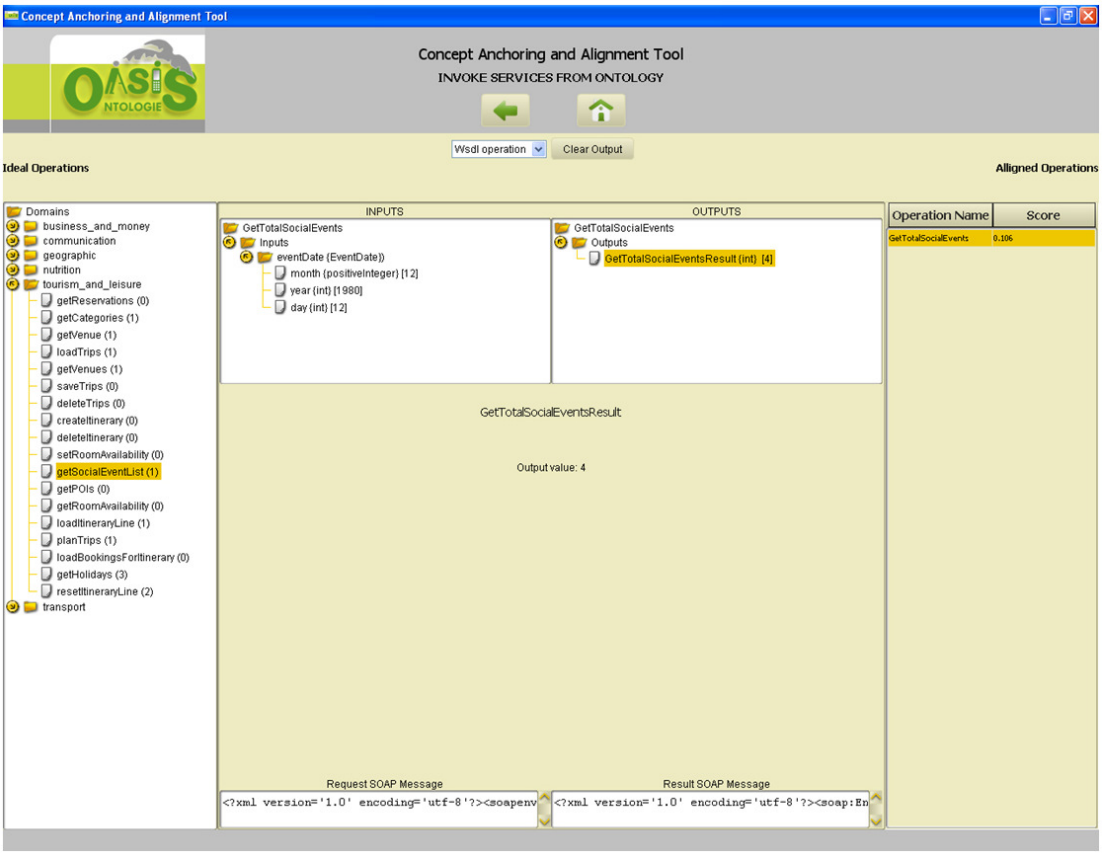


Figure 18: The results that are returned after the invocation of the real web service operation are shown both in the form of a SOAP message or by their content whenever this is possible.

3.3.1. Special Cases

Arrays of elements

Arrays of elements can be defined in an input of a WSDL file, such that the user can submit an input with multiple values. For example, in Figure 19, the invocator identifies that the object "Stops" consists of an array of **`TripStop`** objects (**`TripStop[]`**), and prompts the user to add as many elements within the array as he wants. In Figure 20, the user has added two additional **`TripStop`** objects to the array.

Enumerations

Some WSDL file definitions include enumerations in some of their inputs, that prohibit users to enter any other value except from the ones that belong to a predefined list of values. For example, the definition of the element **`Densitys`** is restricted to the set of values shown in Figure 21.

In such a case, the invocator identifies the specific values that are defined in the enumeration list and prompts the user to select one of them as potential value to the input parameter. In the invocation frame shown in Figure 22 the user presses the "Select value" button and gets the list of accepted values that are defined in the Enumeration list as it is shown in Figure 23. It is then up to the user to select the appropriate value among the available ones that will be used as an input at the invocation process.

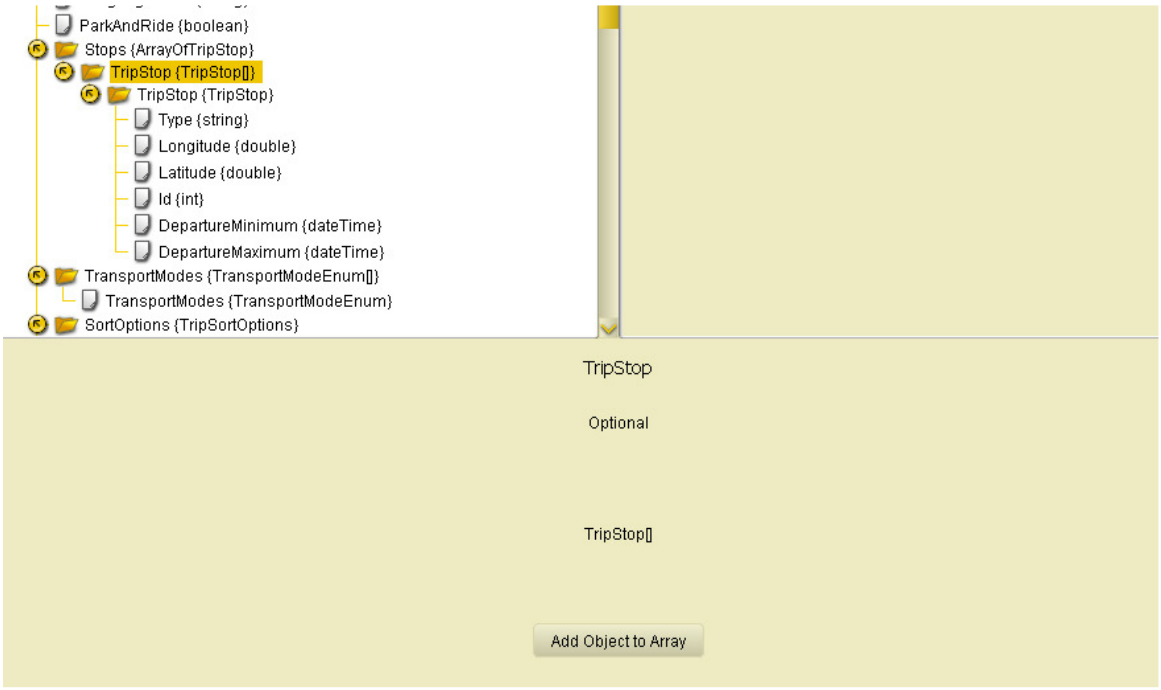


Figure 19: Example of an array element `Stop`. User is prompted to add elements to array.

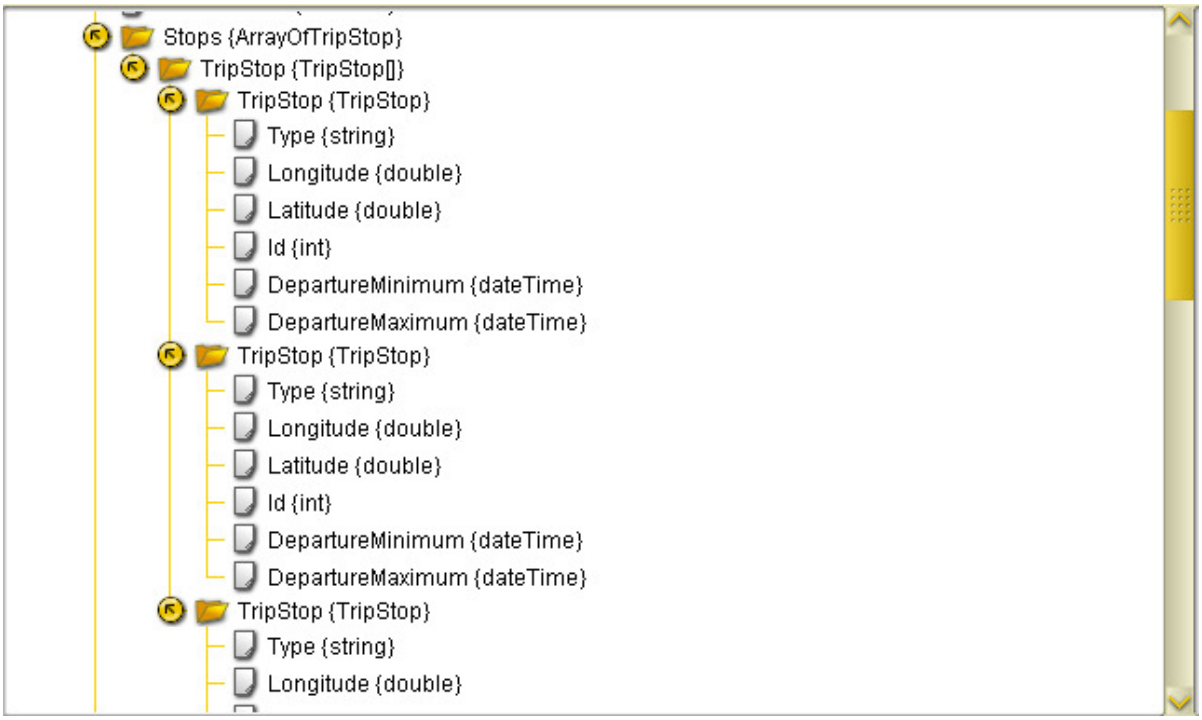


Figure 20: Inputs of the operation when user adds additional elements to array.

```
- <s:simpleType name="Densitys">
  - <s:restriction base="s:string">
      <s:enumeration value="kgPercubicmeter"/>
      <s:enumeration value="lbmPercubicfoot"/>
      <s:enumeration value="lbmPergallonUSliq"/>
      <s:enumeration value="slugPercubicfoot"/>
      <s:enumeration value="gramPercubiccm"/>
      <s:enumeration value="kilogramPerliter"/>
      <s:enumeration value="metrictonPercumeter"/>
      <s:enumeration value="longtonPercubicyard"/>
      <s:enumeration value="lbmPercubicinch"/>
      <s:enumeration value="shorttonPercubicfoot"/>
      <s:enumeration value="kilogramPercubiccm"/>
      <s:enumeration value="aluminum"/>
      <s:enumeration value="iron"/>
      <s:enumeration value="copper"/>
      <s:enumeration value="lead"/>
      <s:enumeration value="gold"/>
      <s:enumeration value="silver"/>
      <s:enumeration value="water"/>
    </s:restriction>
  </s:simpleType>
```

Figure 21: The definition of an element that has *enumeration* values.

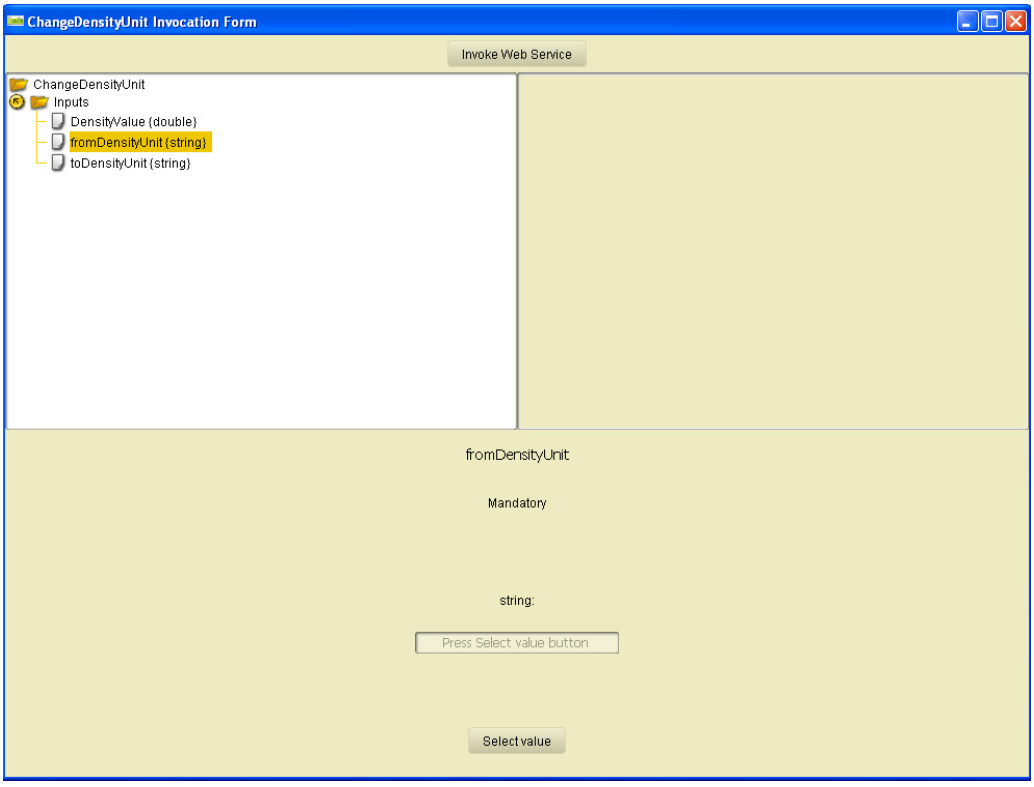


Figure 22: Invocator identifies enumeration values, and allows the user to select specific values from the allowed ones.

3.4. Edit web service metadata

The third option provided on CAAT home screen (Figure 5) is the ability to insert or edit WS metadata that will be used by the AmI platform for personalisation purposes. The AmI requests service invocation through the CCM and receives a list of appropriate real and aligned WS operations that fulfil the particular needs of a use case or an end-user who runs end-user applications on top of the AmI. The AmI agents are then responsible to evaluate the list of appropriate WS operations by ranking its contents based on a set of personal criteria. The purpose of this ranking is the selection of the operations that better matches the user profile or other personalised user-specific preferences. For this ranking additional information is needed. This additional information is provided by the service providers using the "Edit aligned services' panel (Figure 24) of the CAAT. This step is optional but it facilitates the operation selection process on behalf of the AmI and serves personalisation capabilities.

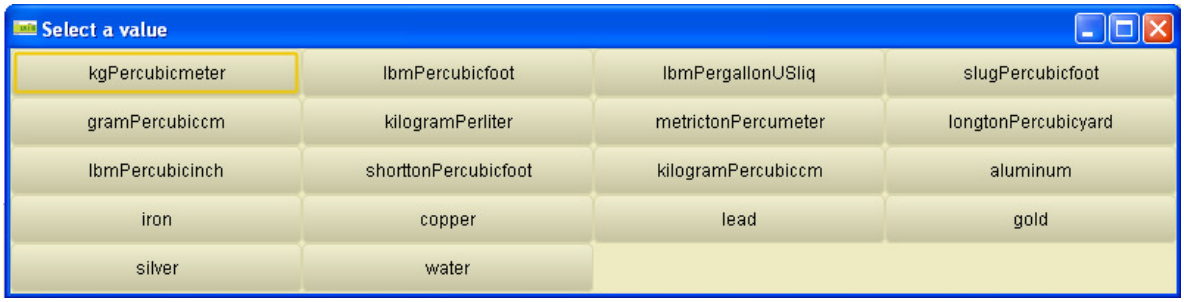


Figure 23: The list of allowed values defined in the Enumeration.

Table 1 lists all metadata parameters that the user is able to fill-in by using a series of forms such as the one depicted in Figure 24.

Table 1. Meta-data parameters

Category	Meta-data parameter	Meta-data parameter description
Common	Age Category	Does the web service support *young elderly*, *elderly* or/and *old elderly*?
	Sex	Does the web service support male/female?
	Language	Which languages does the web service support?
	Impaired Category	Does the web service support OASIS user's impaired category?
	Health Status	Does the web service support permanent/temporary psychological and health status?
	City	Does the web service support OASIS user's current city?
	Countries	List of countries that are supported by web service

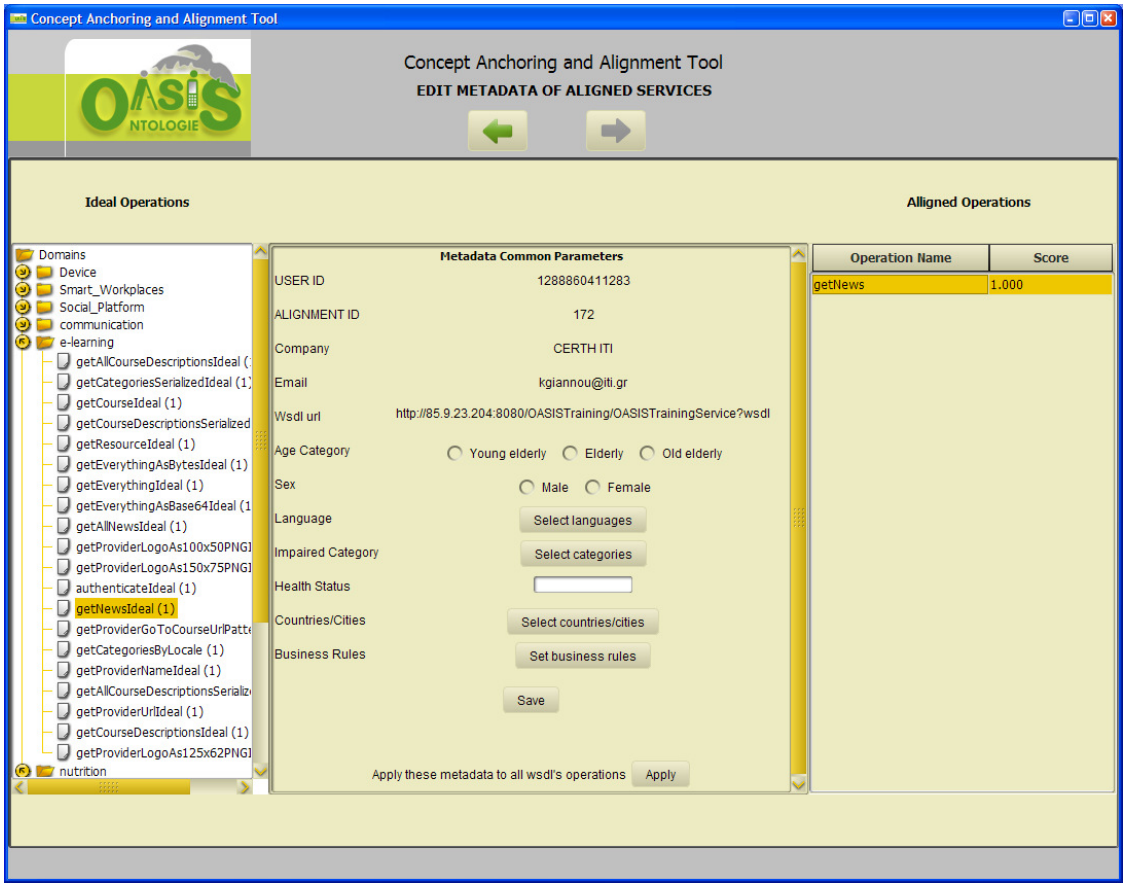


Figure 24. By the third option the user may insert metadata to describe web service properties. These are used for personalisation purposes.

Any service provider can edit the parameters for the selected operation and save them to the registry, by corresponding forms as the ones that are illustrated in Figure 25. Among the various options that are available, the Business Rules properties are described in detail in Section 5 (*Creating Business Rules*).

3.5. Edit services

The fourth option provided in the introduction window shown in Figure 5 is the ability to view or delete all user's aligned web services. By pressing "Next" in the main screen, CAAT parses all user's aligned WSDL operations (this can be a time consuming process) and presents them as it is shown in Figure 26. It should be noted that if during the time of parsing the WSDL operations, a WSDL URL does not respond, it will not appear in the aligned operations tree.

The user has the option to delete an alignment of an operation, or the aligned operations of a WSDL file as in Figure 27. Additionally, he can test the invocation of any of his aligned service by pressing the "Test Invocation" button.

It should be mentioned that by deleting an alignment, the service registry and OR are updated by deleting the appropriate entries and ontology files correspondingly.

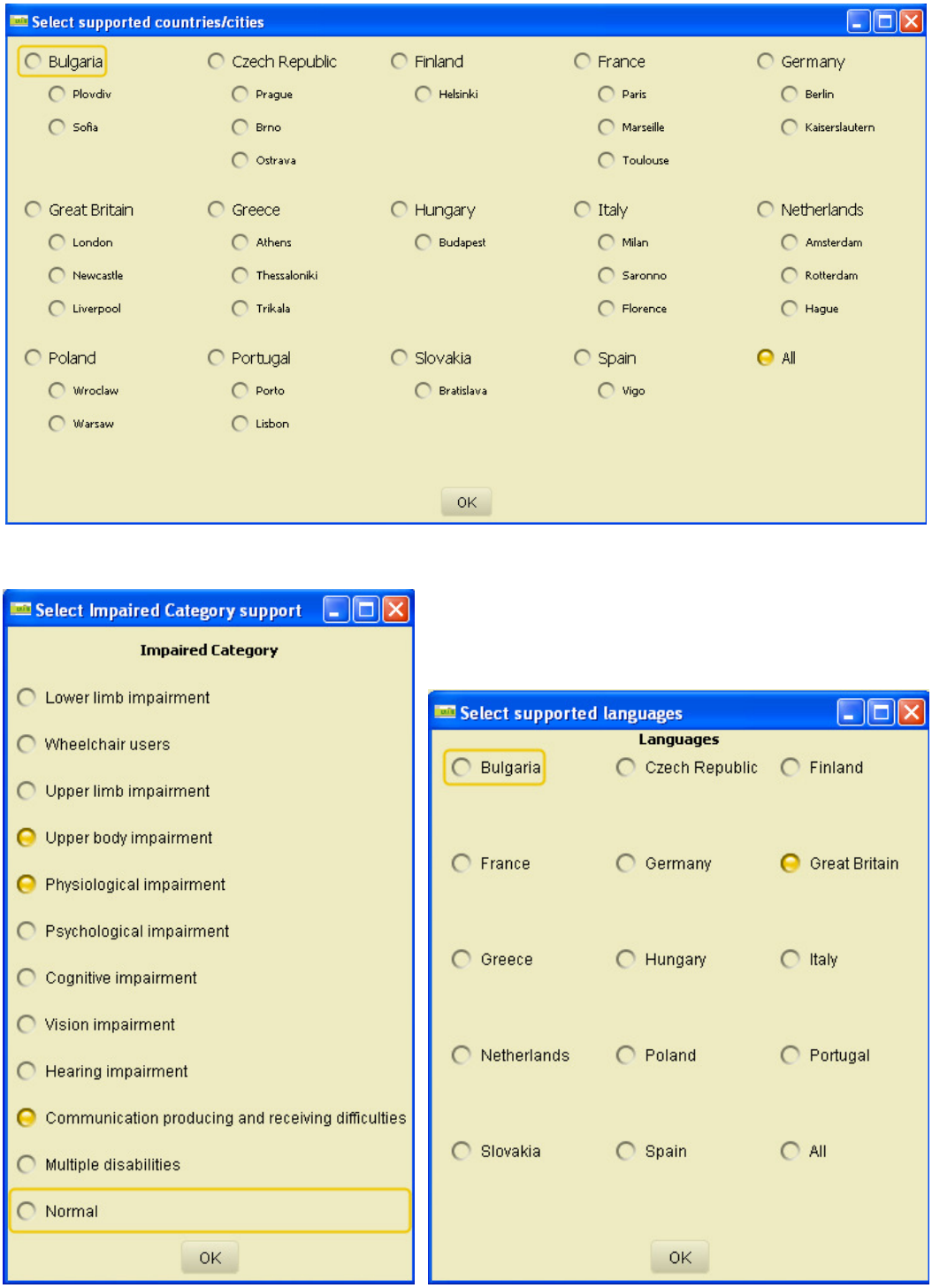


Figure 25: Various forms for inserting metadata parameters.

3.6. CAAT background processes

CAAT has some processes that run in the background in order to provide synchronization between the local copy of the ontology and the ORATE repository, and to provide increased performance when trying to parse already aligned services.

More specifically, a background process ensures that the local copy of the ontology (i.e., loaded ontology in the current CAAT instance) is always synchronized with the one stored in ORATE. Therefore, if an ontology file is added or deleted to ORATE by another provider, a warning message appears to CAAT, prompting the user to update

(synchronize) his local ontology. Service providers are **highly** recommended to update their ontology in such cases, in order to avoid any potential mismatches.

Another background process is parsing all aligned WSDL services, in order to gain performance when provider requests to view any aligned services.

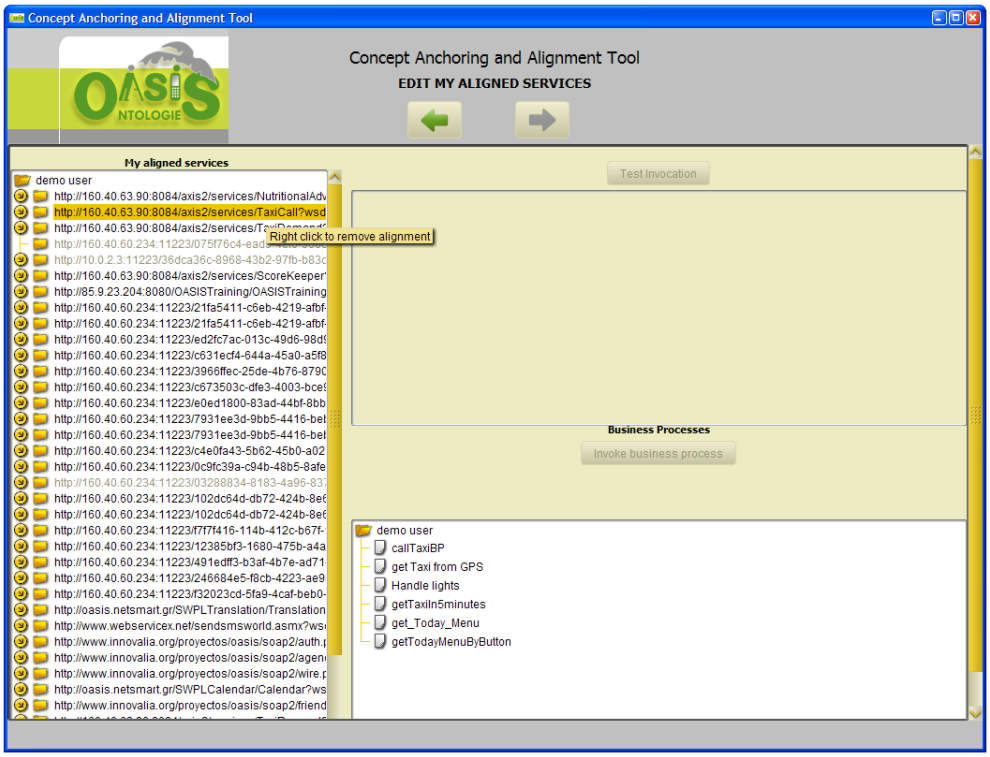


Figure 26: The aligned operations panel.

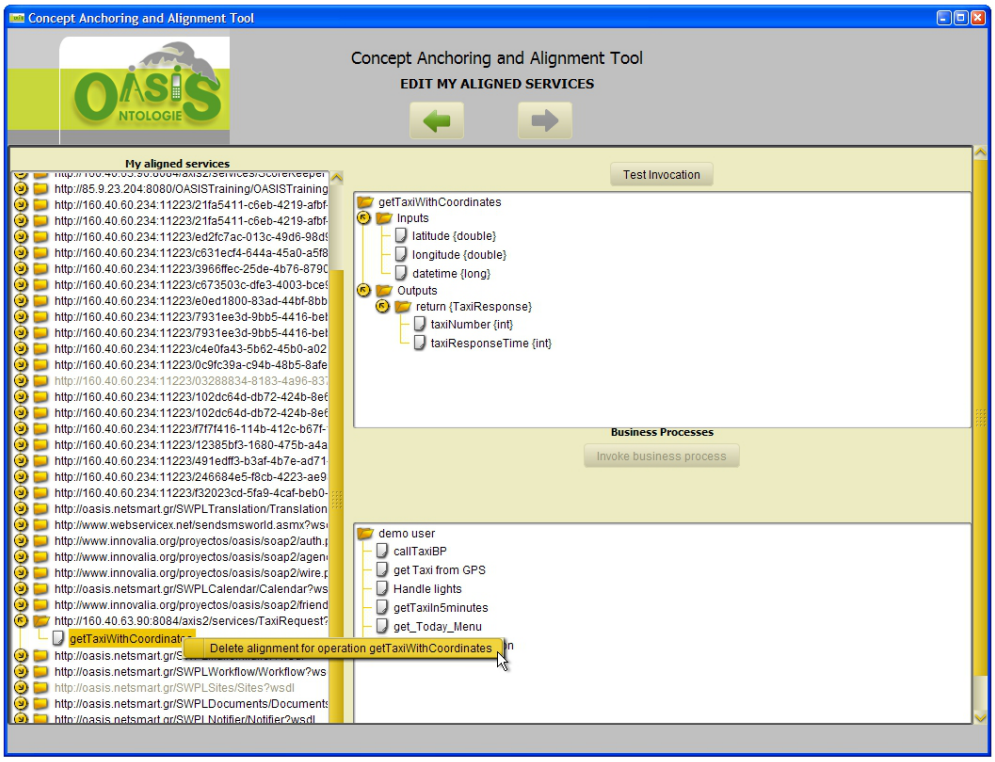


Figure 27: The service provider has the option to invoke his aligned operations, or to delete his aligned web services.

4. Creating and training new classification models

In the special case where the service that an SP wants to integrate in the OASIS platform does not belong to any of the provided domains, the user should create and train a new domain with available data. The creation of a new domain has therefore some prerequisites that the service provider has to consider. The WSDL file that wants to integrate to the new domain, should have at least two operations in order to build and train classification models at the 2nd classification step. Additionally, the service provider should have to submit local WSDL files that represent the domain and will be used as training data for classification purposes at the 1st classification step.

4.1. Create a new application domain

To create a new application domain, click on the "Change Domain" Button on the panel that appears after the completion of the first level classification process that is described in Section 3.2 (Figure 28). On the drop-down menu that appears select the ***<Create New Domain>*** option. The pane depicted in Figure 29 appears.

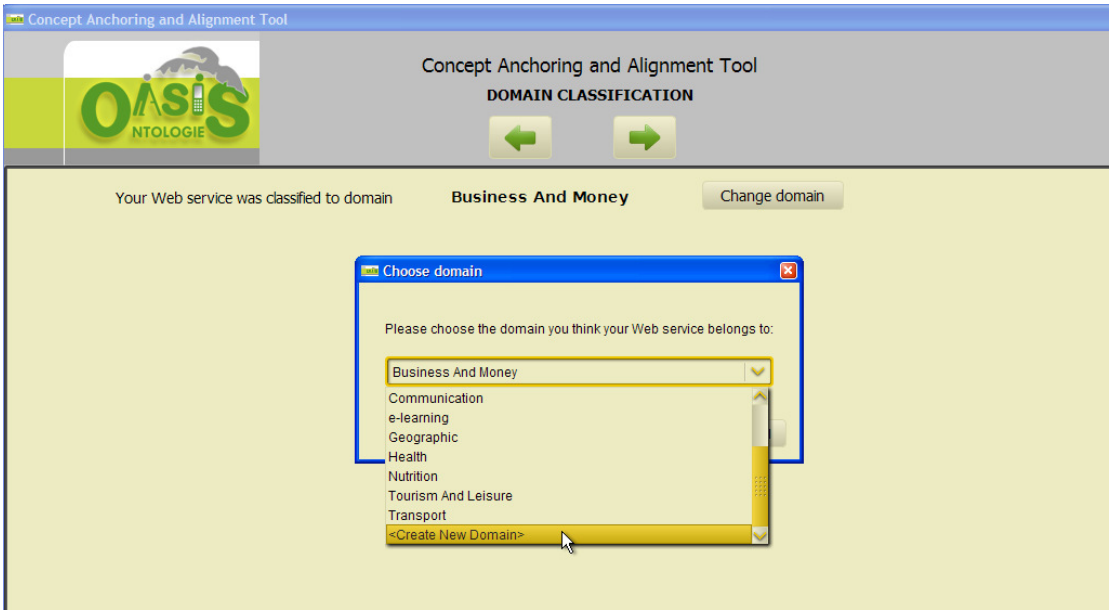


Figure 28. By pressing the Change Domain button the user may select to create a new domain

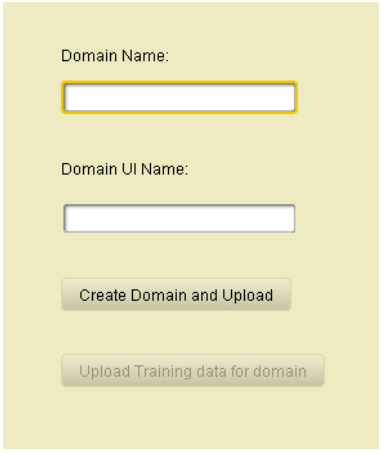


Figure 29. New Domain Name and description option.

Directions are given in what follows for the creation of a new domain. In the *Domain Name* field write a unique short name identifier (e.g. "edu") that will describe the ontology in the repository, while in the *Domain Description* field write the name of the Domain as you would like it to appear in the tool and to the end user (e.g. "*Education*").

Press the "*Create Domain and Upload*" button in order to create the new domain in the form of an ontology that will be uploaded on ORATE. As soon as the ontology will be uploaded on the ORATE repository, a confirmation message as the one shown in Figure 30 shows up. However, it takes some time for the ontology to appear in the ORATE website (http://orate.iti.gr/).

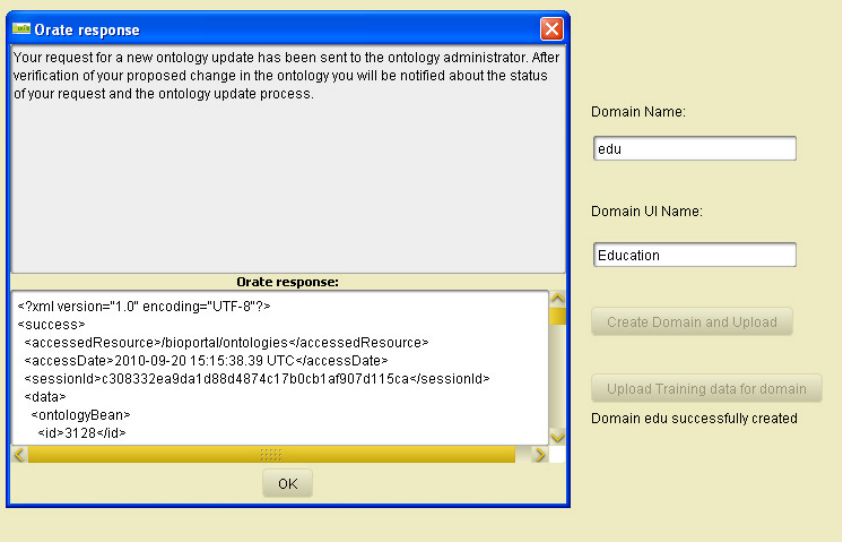


Figure 30. As soon as the ontology is uploaded on the ORATE repository, a confirmation message shows up. This means that the ontology was been successfully uploaded.

⚠ **Attention**: When you create a new domain ontology in the way described so far, this new ontology is automatically uploaded on the ORATE repository (http://orate.iti.gr), where it appears as a new ontology under the Pending Approval category. The new domain will be only available after the ORATE administrator will finally approve the new ontological domain. If the ORATE website for any reason is not operational, this step cannot be completed.

After the ontology has been created and uploaded in ORATE the user presses the "Upload Training data for domain" button (Figure 31) and, in order to upload a set of local domain-representative WSDL files that can be used as training data for the new domain. A file selection dialogue window appears (Figure 32) that allows the user to select the WSDL files, and upload them to the training data repository. When they are uploaded, the user is ready to press "Next" in order to proceed.

By pressing "Next", some background processes are performed that are essential for the creation of the new domain. More specifically, equal number of ideal operations to the number of the wsdl operations are created and uploaded to Orate. Additionally, equal number of real operations will be created as annotated ontology files to the created ideal operations in order to be the training data for the 2nd classification step.

This may be a time consuming process, but it is essential in order to have an automatic way of creating new application domains.

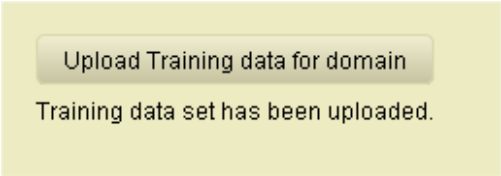


Figure 31: The user presses the "upload training data for domain" button in order to provide the web services to be used as training data for training the new domain.

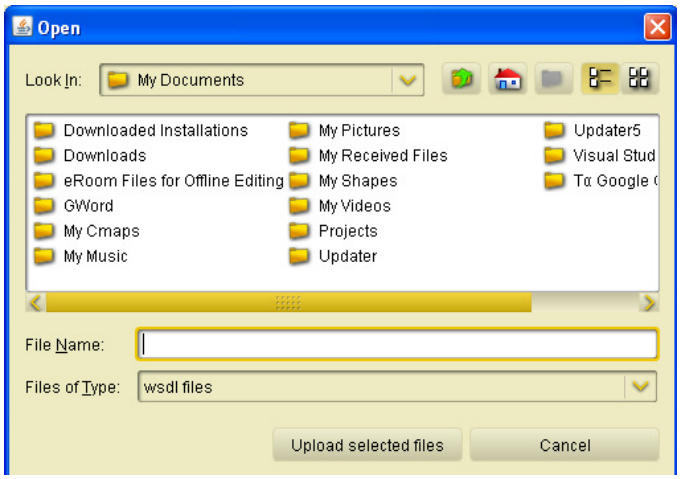


Figure 32: The user is prompted to upload wsdl files to the training data repository

After uploading the ontology files and training the necessary models, the tool shows the 3rd classification step (Figure 33), where the user can view the created ideal operations in the newly created domain. In order for an alignment to be saved, firstly the domain has to be reviewed and approved by an administrator.

⚠ **Attention**: In order for the training process to be successful and the newly created classifier to support the new domain, the ontology administrator should approve the ontology in ORATE and move it from the "Pending Approval" to the "Web Services" category. This is required before proceeding to the next step.

The following process is required for manually training a new classifier that corresponds to a newly created domain. In order to start training, go to the initial panel that is shown in Figure 34 and press the "Create New Classification Model". The extended panel that appears in Figure 34 allows you to initiate the training process. This involves the following steps in this order:

1. ***Data preparation***. To start the data preparation stage press the "Pre-Process Data", after selecting the appropriate parameters or leaving the default choices *(more details on the classification parameters, can be found in project public deliverable D1.3.2, "Data Content Connector")*
2. ***Training***. After the previous step is completed, you may select an appropriate classification algorithm (e.g. NaiveBayes) and start the training process by pressing the "Train" button.

3. ***Saving Model***. Once the training process is completed you may save the newly created model. This is named automatically based on the current date and time. The name of the newly trained classifier that supports the new domain appears on the list of the Available Classification model.

From this point on, the service classification procedure is performed as described in Section 3.

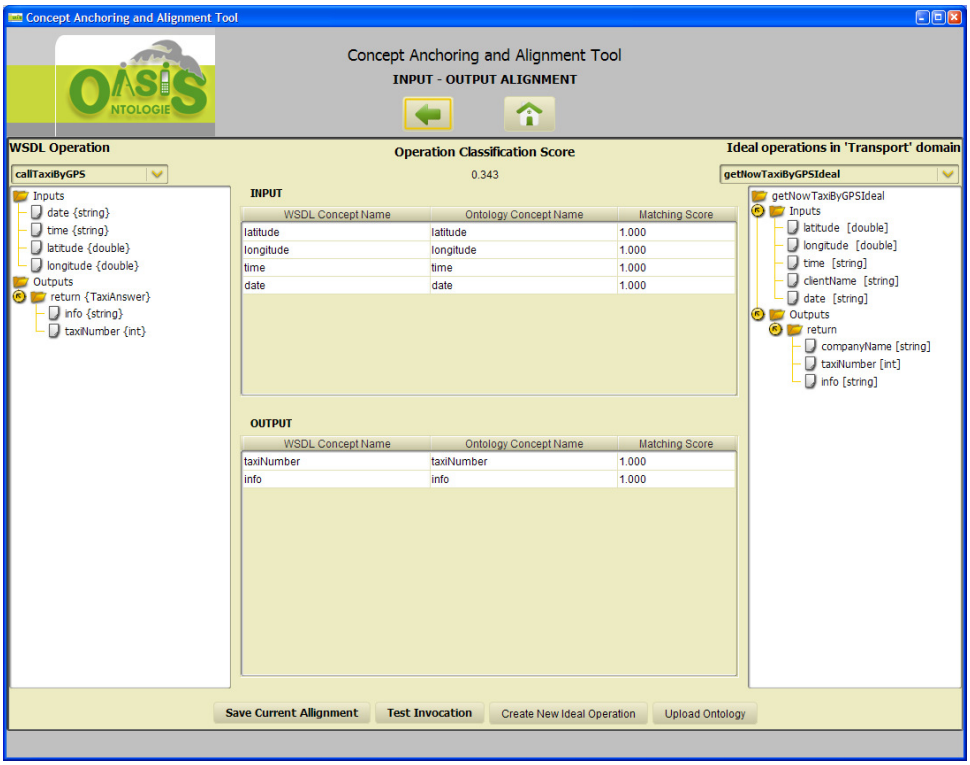


Figure 33: When creating a new domain, equal number of ideal operations to the wsdl operations are created, but no alignment can be saved before the domain approval by the administrator.

4.2. Service Ontology

In order to annotate the various elements in a WS, you can create various service ontologies similar to the official Service Ontology ("***ProfileHierarchy.owl***") that is stored in the ORATE repository under the Web Services category and it is loaded each time CAAT starts. Any new ontology that is manually created to describe a service and its domain ***should adhere to the structure of the ProfileHierarchy.owl***. A snapshot of this ontology is shown in Figure 35 (a).

The main structural elements of the service ontology skeleton that forms a template for the description of any service in the context of OASIS are the following:

- Class *serviceDomain*. This class describes the domains in which a WS is classified in the first level.
- Class *operation*. This class describes a WS operation, used for the second classification level.
- Class *Concept*. This class describes various I/O operation parameters.
- Object properties *hasInput*, *hasOutput*, *hasServiceDomain*, *hasType*, *belongsToPrototype*. These properties are required for the description of the input, output and the application domain to which an operation belongs. In addition, the property belongsToPrototype is used to indicate if the current

operation is a real or an ideal one. These properties are defined in Manchester syntax as shown in Figure 35 (b).

- Datatype properties *hasName*, *hasPreferredUIName*. These properties are required for the description of the operation and its inputs and outputs.

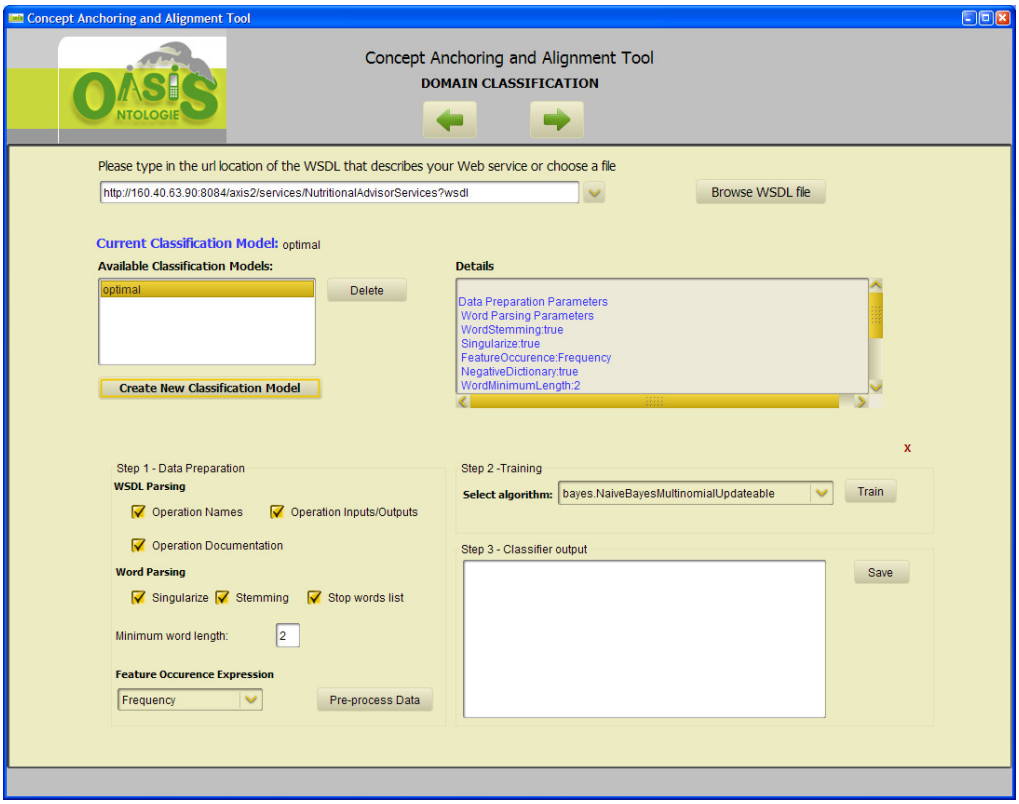
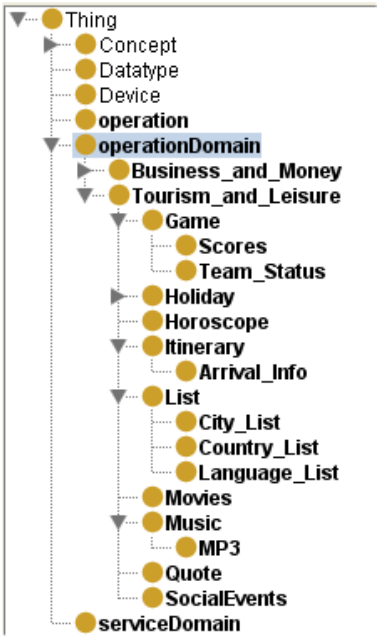


Figure 34. Training a first-level classification model.



(a)

ObjectProperty: **hasInput**
Domain: **Operation**
Range: **Concept**

ObjectProperty: **hasOutput**
Domain: **Operation**
Range: **Concept**

ObjectProperty: **hasServiceDomain**
Domain: **Operation**
Range: **Concept**

ObjectProperty: **hasType**
Domain: **Concept**
Range: **Concept** or **DataType**

ObjectProperty: **belongsToPrototype**
Domain: **operation**
Range: **operation**

(b)

Figure 35. (a) Snapshot of the Web-service ontology in Protégé, (b) some of the most important object properties in Manchester notation

4.2.1. Manual Creation of a new Domain

The Service Ontology supports the creation of a new domain. In order to add a new domain, select the class "serviceDomain" in Class browser of Protégé editor and create a new instance to this class in the *Instance browser*. Selecting this new instance brings up the *Individual Editor* (Figure 36), where there are two datatype properties that should be filled in manually with useful information from the user. The first property "hasPreferredUIName" shows the name of the new domain added as it will appear in the user interface of CAAT. The second property "categoryName" provides information about the category name of the new domain added.

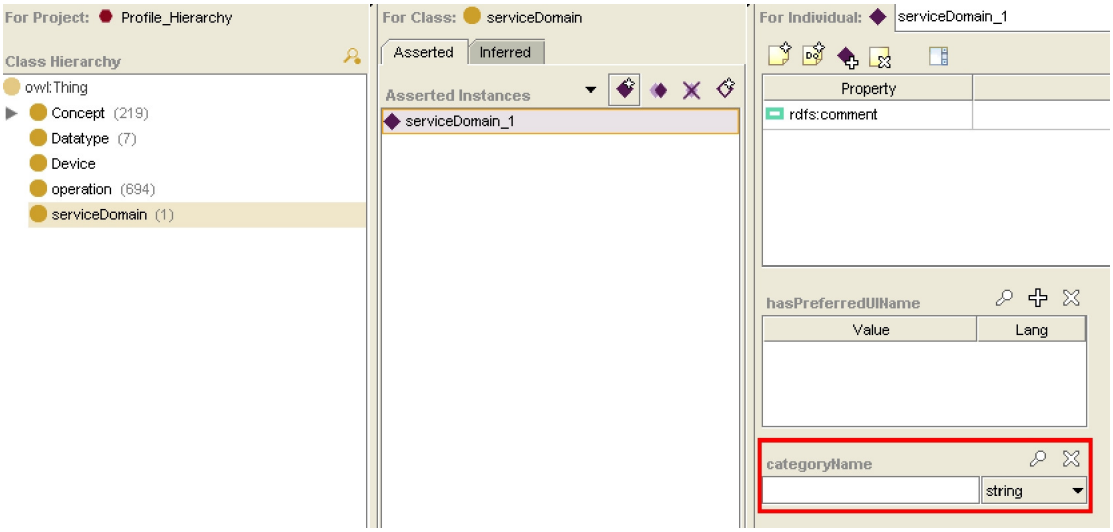


Figure 36. Add new domain in the Service Ontology using the Protégé ontology authoring tool.

4.3. *Create a new "ideal" operation*

When a web service operation does not match semantically with any of the existing ideal operations, a new one should be created. Ideal operations are abstract ontological descriptions that describe the semantics of the corresponding real operations that are included in WSDL files. There are two ways to create an ideal operation: i) within CAAT and ii) by manually creating a new Service Ontology.

4.3.1. Create a new ideal operation in CAAT

CAAT provides through its user interface the ability to create a new ideal operation. This is required in two circumstances: a) when a new domain has been created and new ideal operations should be defined in the new domain, b) a web service cannot be classified correctly in one of the existing ideal operations of an existing domain therefore a new ideal operation should be created to semantically describe the current service.

Before creating a new ideal operation you should perform a 2nd and 3rd level classification of a newly inserted web service, as described in Section 0. To create a new ideal operation that corresponds to a particular real operation, select the real operation for which you want to create the new ideal operation and press the "Create

New Ideal Operation" button that appears on the 2nd and 3rd level classification results panel (Figure 37). A new ideal operation appears on the right panel with name

`*Ideal`, where **`*`** is the name of the real operation. In the example shown in Figure 37 we create the **`getDinnerIdeal`** operation for the **`getDinner`** real operation. The new ideal operation is created as a copycat of the corresponding real one. You may further modify the newly created ideal operation by using the context sensitive menu that appears by right clicking on the ideal operation tree. The appearing menu gives the possibility to modify the name of the ideal operation and to add, delete or modify existing ontology concepts (I/O in the ideal operation).

In order for the newly ideal operation to be functional, you should publish it in the ORATE ontology repository by clicking on the "Upload Ontology" button. This creates automatically a new Service Ontology that is uploaded on ORATE under the "Pending Approval" category. It is up to the ontology administrator to approve the new Service Ontology (i.e. to move it from the "Pending Approval" to the "Web Services" category and make it operational for all service providers in CAAT).

It should be noted that the name of the ontology operation should be unique to the hyperontology, and for this reason if the user tries to submit an already used ideal operation name, the tool will ask for an alternative.

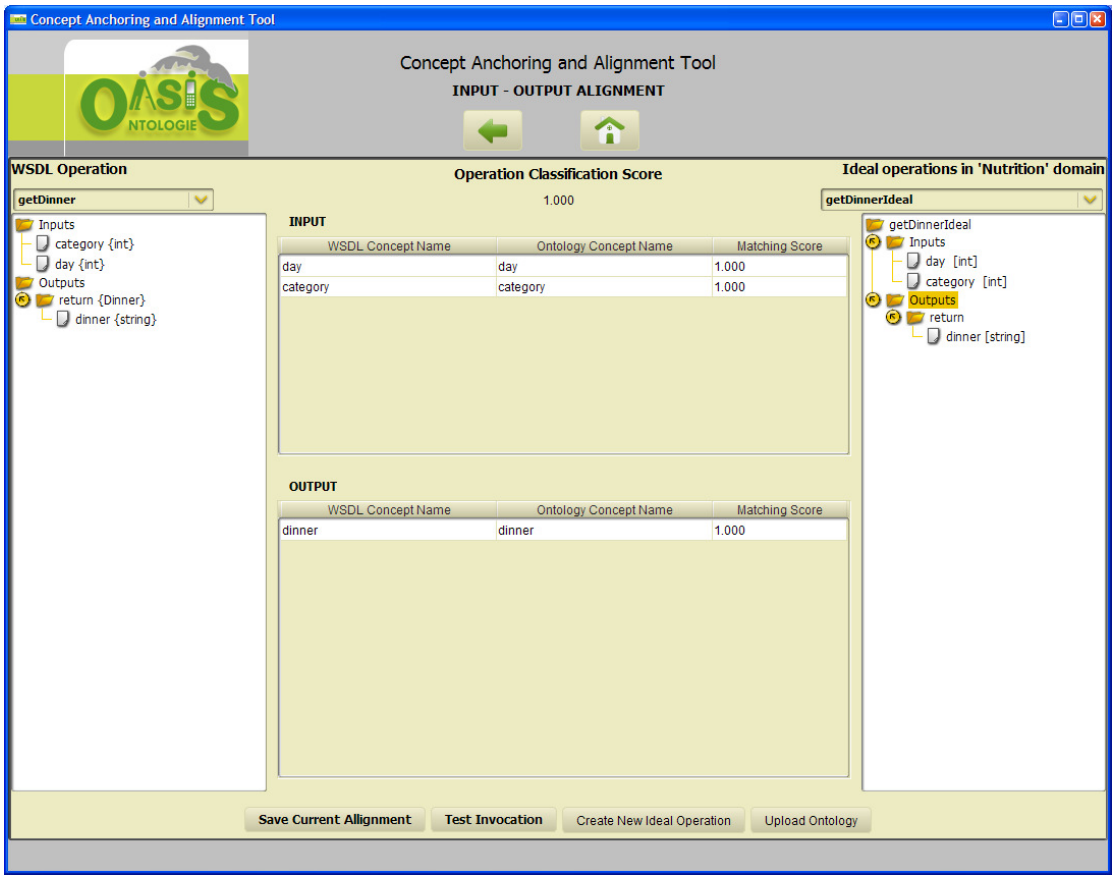


Figure 37. Create a new ideal operation using the CAAT user interface.

4.3.2. Add a new ideal operation in the Service Ontology

We assume that a new Service Ontology has been created as described in Section 5.2. To add a new ideal operation, select the class "**Operation**" in *Class browser* of the Protégé ontology authoring tool and create a new instance to this class in the *Instance browser*. Selecting this new instance brings up the *Individual Editor* that shows the

following properties that should be filled in manually with appropriate data (Figure 38):

- *hasPreferredUIName*: Shows the name of the new ideal operation added as it will appear in the user interface of CAAT
- *isPrototype*: This property should be **true** for every ideal operation.
- *belongsToPrototype*: This property should remain *null*, since the operation that is being added is *ideal* (as opposed to a *real* one).
- *hasInput*: required for the description of the inputs of the ideal operation. This property's range is the class "Concept", thus any instance under this class and its subclasses are potential inputs of the ideal operation. In order to add a new class to describe the inputs of the ideal operation consult Section 5.3.3.
- *hasOutput*: required for the description of the outputs of the ideal operation. This property's range is the class "Concept", thus any instances under this class and its subclasses may be potential output parameters of the current ideal operation. In order to add a new class to describe more outputs of the ideal operation consult Section 5.3.3.
- *hasServiceDomain*: required for the description of the service domain to which the ideal operation belongs. To assign a particular domain to this operation select an instance of the class "serviceDomain".

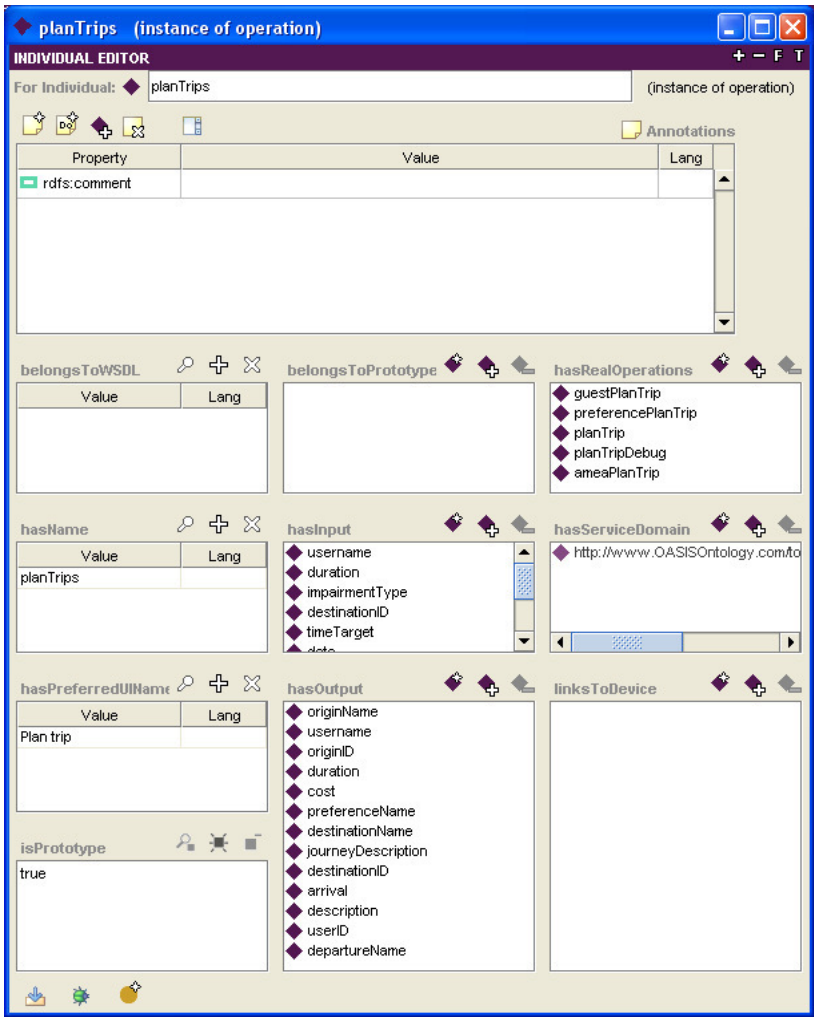


Figure 38. Creating a new ideal operation in Protégé

4.3.3. Add new concept classes that describe the I/O of the operation

In case the existing instances in class "Concept" and its subclasses are not sufficient to describe the inputs and outputs of the ideal operation, you may create new classes under the class "Concept". For each new class, new instances should be added and for these instances the following properties should be filled manually on Protégé *Individual editor* (Figure 39):

- *hasName*: Shows the name of the input or output of the operation.
- *hasType*: Shows the type of the input or output. It could be another I/O or an instance from class "datatype" (string, float, Boolean, etc.).
- *isArray*: true or false according to the web service description of I/O. If the I/O is array, this property should be true.

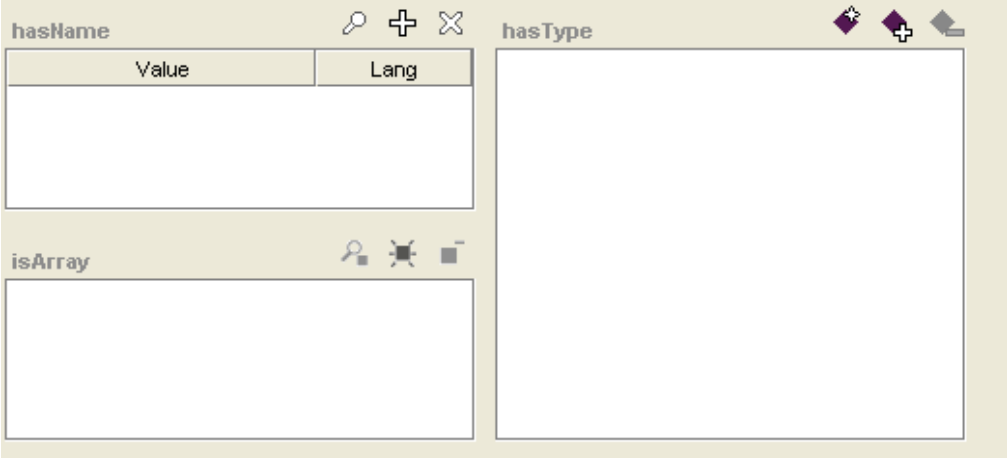


Figure 39. Properties that contain information about I/O of operations

⚠ **Attention:** In all cases where you create a new Service Ontology in manual mode, you should upload the new Service Ontology on ORATE in order to make it visible in CAAT. If the ontology needs further approval by the ontology administrator it should be placed under the "*Pending Approval*" category. When the ontology is placed under the "*Web Services*" category it is visible in CAAT.

4.3.4. Training

After the training data has been created as described in the previous section, the training process can be instantiated. The purpose of training is to create a new classifier in a way similar to the one adopted for first-level classification process that will support the new ideal operations. A trained classifier will be able to classify the operations of any real arbitrary service to the ideal operations that are defined in the ontology. Training can be performed after a WS has been classified in a particular domain (first level classification) and before the 2nd and 3rd classification occur. By Pressing the "*Create New Classification Model*" button on the Selection classification algorithm panel after the 1st level classification is completed, the panel is extended as shown in Figure 40. This panel provided by the tool in order to allow selection of the appropriate combination of parameters that are related to the creation of a new model. It provides a set of different sub-panels for data preparation and training. This functionality is provided to the users as an optional feature and it especially exists for experimentation purposes with different parameters since there is practical evidence

that the modification of these parameters can affect the quality of the classification method.

Similarly to the first level classification training process the training process is completed in three steps: data preparation, training and saving of the created model. The training process is be executed based upon the training dataset created in the previous step. The result of the training process is the creation of the new classification model that takes into account the new ideal operation(s) that have been previously created. You can create as many classification models as you want but each time only one can be selected for execution. After the completion of the training process and the creation of the new classification model (classifier) some useful statistics are shown for this model in the corresponding text area.

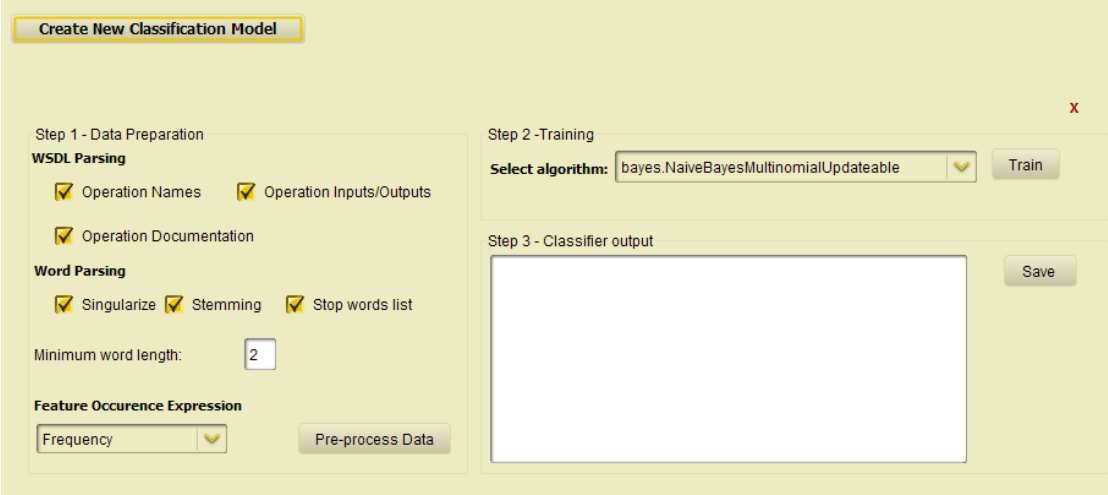


Figure 40. Create new classification model panel. The user may select the parameters of the data preparation and training processes that lead to an appropriate classifier.

5. Creating Business Rules

5.1. Supported Business Rules

From version 2.0 and on, CAAT supports the creation and execution of business rules. According to the business rules that are defined within the ontology, web service providers are able to define which web service operations will be free of charge and which ones will be payable (providing also the corresponding cost for them).

In addition to this, the web service providers can specify which service providers' services are allowed to be chained to their own services through a service cascading scenario (see previous chapter). This will be made available during the service alignment process, and in particular at the 3rd level classification process, which involves matching between input/output parameters of web service operations. The payable operations will be visible to all end-users, however only those users who have paid for particular web services will be able to invoke the corresponding operations. Those operations, for which no specific business rule is applied are assumed to be free of charge.

Business rules can be also defined for individual operation's outputs. The service provider can determine which web service operations outputs will be available for free viewing after the invocation of the particular service and which ones will be available at a specific cost. Moreover, service providers may define IPR regarding the participation of their web services in a service chain that is composed of multiple simple services by different service providers. In a typical web service cascading scenario (see Section 6), a composite service *z* may be defined from two (or more) single services *x*, *y* where service *x* takes service *y*'s output and uses it as an input. For example, web service provider *A* may define a business rule, according to which only those services that are provided by providers *A* and *B* are allowed to use *A*'s services output as their input.

Table 2 presents the specific business rules that are defined for both levels. In particular, the *hasCost* property determines whether the specific operation or output parameter will be provided at a specified cost, which is expressed as a decimal number ("*double"*) with additional information, such as the currency (e.g. euro, US dollar, etc.) and the potential frequency of payment, i.e. if payment is imposed per invocation, or on a timely fashion, etc. The property *freeAllowedNoOfInvocations* is equal to the number of times that the specific operation is allowed to be launched free of charge. Similarly, the *freeAllowedDaysOfInvocation* determines the time (in days) that the operations may be invoked free of charge (e.g. one month trial use).

The remaining properties are applied to the operation's outputs level. Property *allowCascading* determines whether operation's outputs are allowed to participate in cascading configurations (as e.g. the outputs o_3 and o_4 of the *y* operation illustrated in Figure 40), while *allowedWSDLsForCascading* determines the specific web services (in terms of the location of their WSDL files) that are allowed to access this operation's output parameter for cascading purposes. Finally, the property *hasCostForCascading* is used to define the amount of money that is charged each time the specific output is used for cascading. Additional parameters that are defined for this property are similar to the ones defined for the *hasCost* property.

Table 2: Specific Business Rules for web services.

Property	Type
hasCost	- Cost: double - Currency: String (EUR,USD,GBP,etc.) - Payment charge: String with possible values: Per invocation Per month Per year Unlimited
freeAllowedNoOfInvocations	int (only if *hasCost*>0)
freeAllowedDaysOfInvocation	int (only if *hasCost*>0. Counting is started at first invocation)
allowCascading	Boolean
allowedWSDLsForCascading	Array of Strings (only if *allowCascading*=true. If no values are specified all WSDLs are allowed for cascading use)
hasCostForCascading (cost for using this operation's outputs for cascading)	(Only if *allowCascading*=true) - Cost: double - Currency: String - Payment charge: String with possible values: Per invocation Per month Per year Unlimited

After the alignment process, the business properties are described within the web service ontology file (OWL) which is uploaded to the ontology repository (ORATE), and after verification the new web service with these rules is available for invocation by the end-user. User profile will hold information about end-user's payments and transactions on web services operations.

5.2. Defining business rules in CAAT

The capability to define business rules in the ontology has been added to the CAAT implementation. A provider may insert business rules to his aligned operation either by right click on the operation tree at the third classification step, or by the service metadata option as in Figure 41.

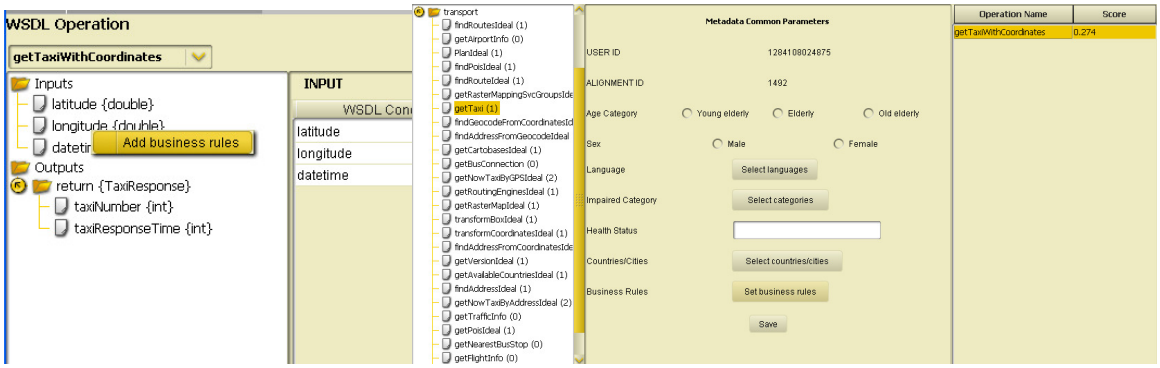


Figure 41: Service Providers may define business rules to his operation at third classification level or at the metadata frame.

In Figure 42, the business rules editor is presented. The service provider can insert all the information described in Table 2 in order to apply the rules to his operation. Moreover, SPs can browse and select those specific aligned operations that are allowed for cascading. Available options to choose from are by domain, service provider or by company, illustrated in the window of Figure 43.

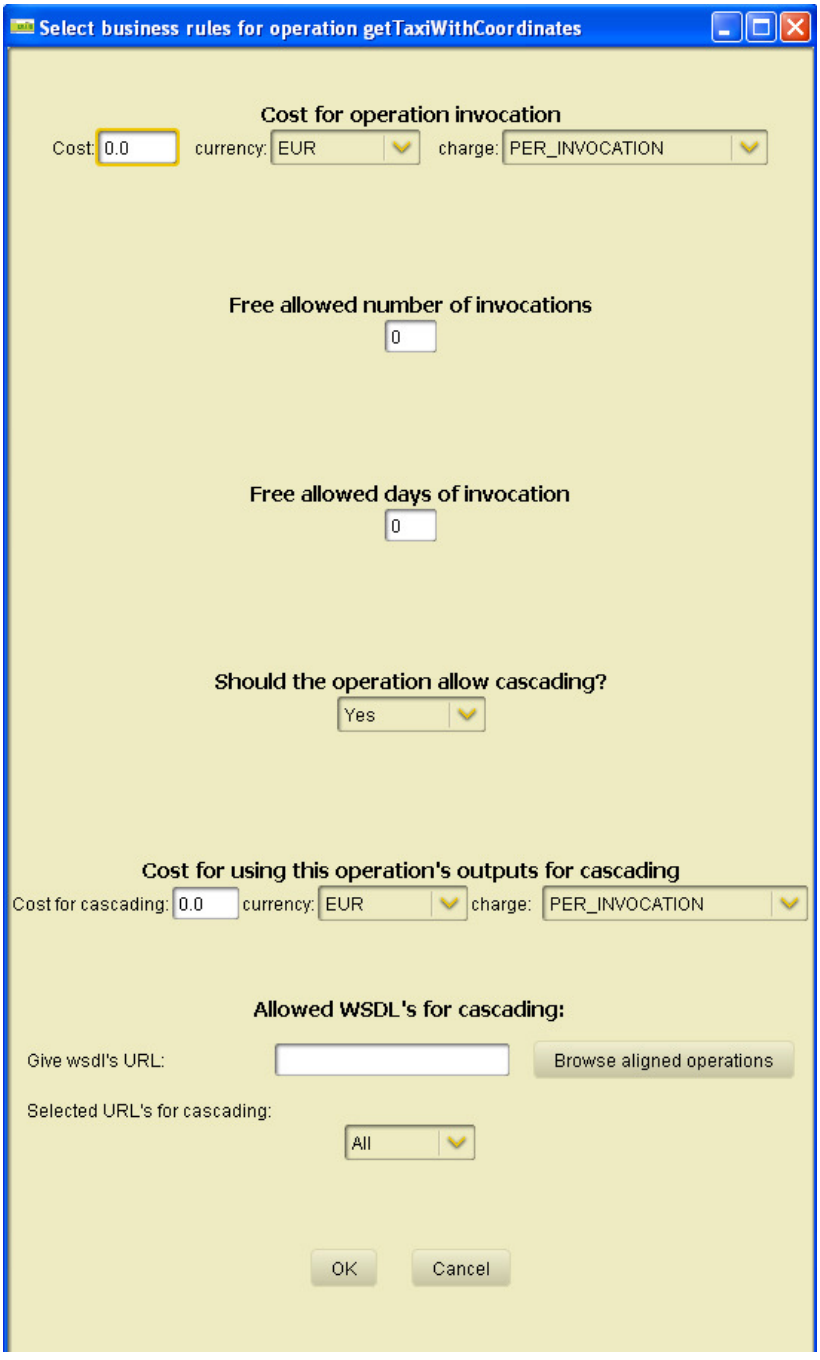


Figure 42: The business rules editor.

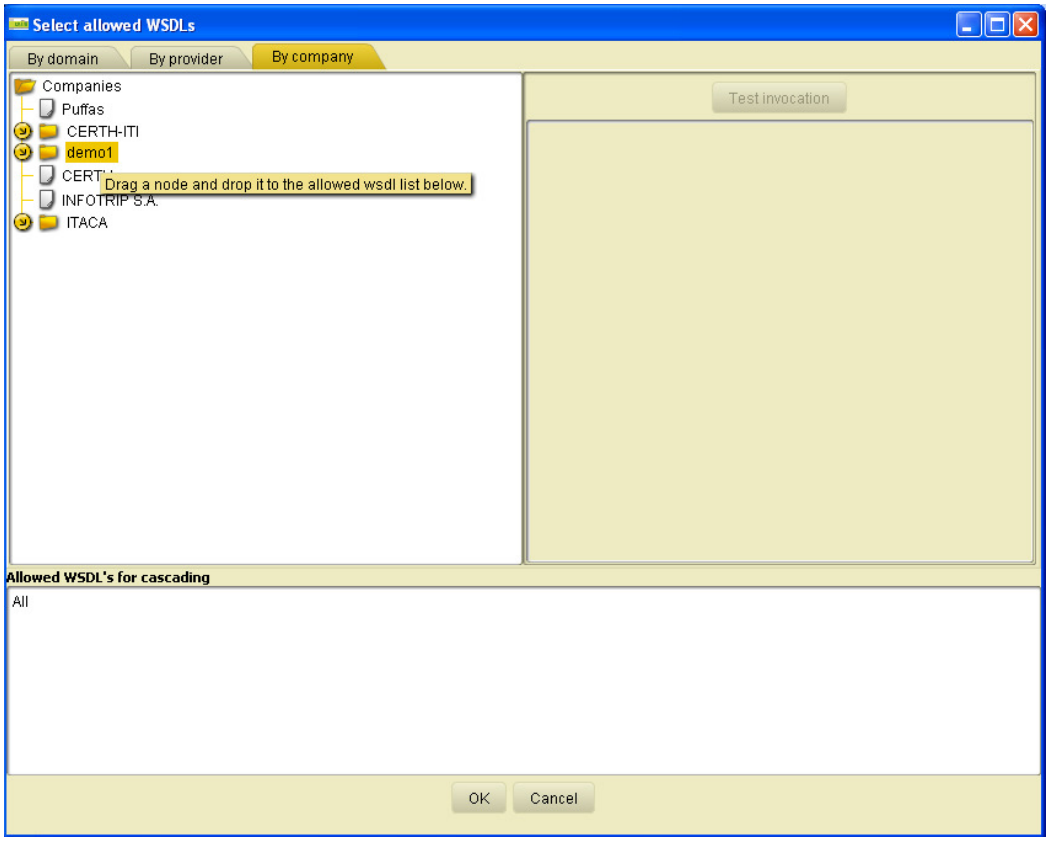


Figure 43: Drag and drop functionality in order to add WSDL files allowed for cascading.

6. Business Processes-Web Service Cascading

6.1. Introduction

The term *business process* is a collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer or customers. It often can be visualized with a flowchart as a sequence of activities [1]. By *web service cascading* we refer to the process of linking web service operations the one next to the other, so that at least one of the interconnected operation's input is provided as an output to another linked operation.

Web service alignment is implemented in REMOTE at the ideal-operation level. By using the new UI components introduced at the 3rd alignment level, service providers are able to explore one domain's ideal-operation outputs in order to match them with the inputs of their service-specific operations. The service providers have the option for automatic or manual match/search within the outputs of domain's ideal operations. The matching process parses all ideal operations of the domain the service was classified to (by default), and performs a composite matching scheme.

More specifically, the matching algorithm consists of three levels of matching: lexicographic, structure and data type matching. Special focus at this alignment is given on lexicographic and data type matching in order to prevent unsuccessful cascaded invocations. The supported UI that is part of CAAT, in addition to the automatic alignment option, also supports manual selection with drag & drop functionality. Furthermore, the ability of exploitation of ideal operations belonging to other domains is supported for generalization purposes.

Once the matching is performed, the aligned real operation with the highest ranking is selected for invocation purposes. Thus, the cascading mechanism is applied between the input parameters of a real operation and the output parameters of another real operation as illustrated in Figure 44.

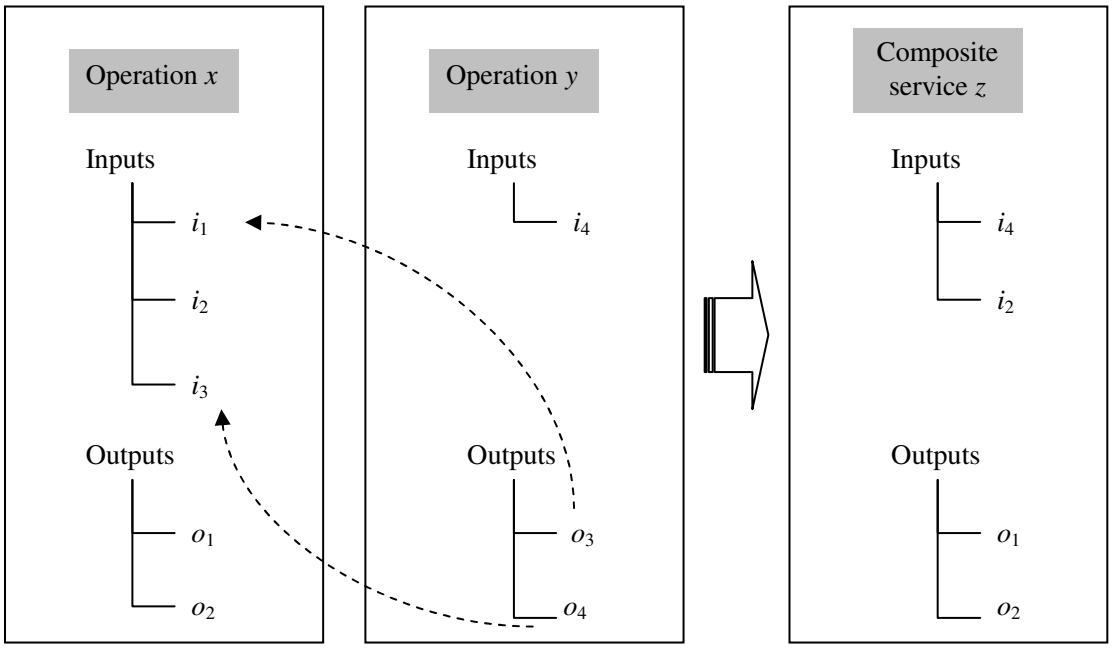


Figure 44. The result of cascading mechanism between two operations *x*, *y*. The aligned pairs (i_1-o_3) and (i_3-o_4) result in a unified cascaded operation with two inputs and two outputs.

In Figure 45, the cascading mechanism is presented at the ideal-operation level. The real operation "*Operation 1*" and its inputs/outputs are aligned through third classification level to the "*Ideal operation 1*". Then, an automatic or manual alignment is performed that results in matching of inputs "*in3*" and "*in4*" with the appropriate ideal operation's outputs (in this example "*out5*" and "*out7*" are selected). In order to make the mechanism more effective in terms of performance, only ideal operations with aligned real ones are investigated for possible matching. To this end an exploration is performed within CCM for aligned operations to ideal operations "*Ideal operation 2*" and "*Ideal operation 3*". These consequently are aligned to inputs "*in3*" and "*in4*" enabling the whole cascading invocation mechanism.

It should be noted that business rules in aligned operations, are inherited through the cascading mechanism. For example, if output "*out7*" in Figure 45 is aligned with an output that is set to be chargeable (e.g. "*out7*"), then for the invocation of Operation 1 the end user has to provide the appropriate payment. More details on CAAT business rules mechanism are provided in the next Chapter.

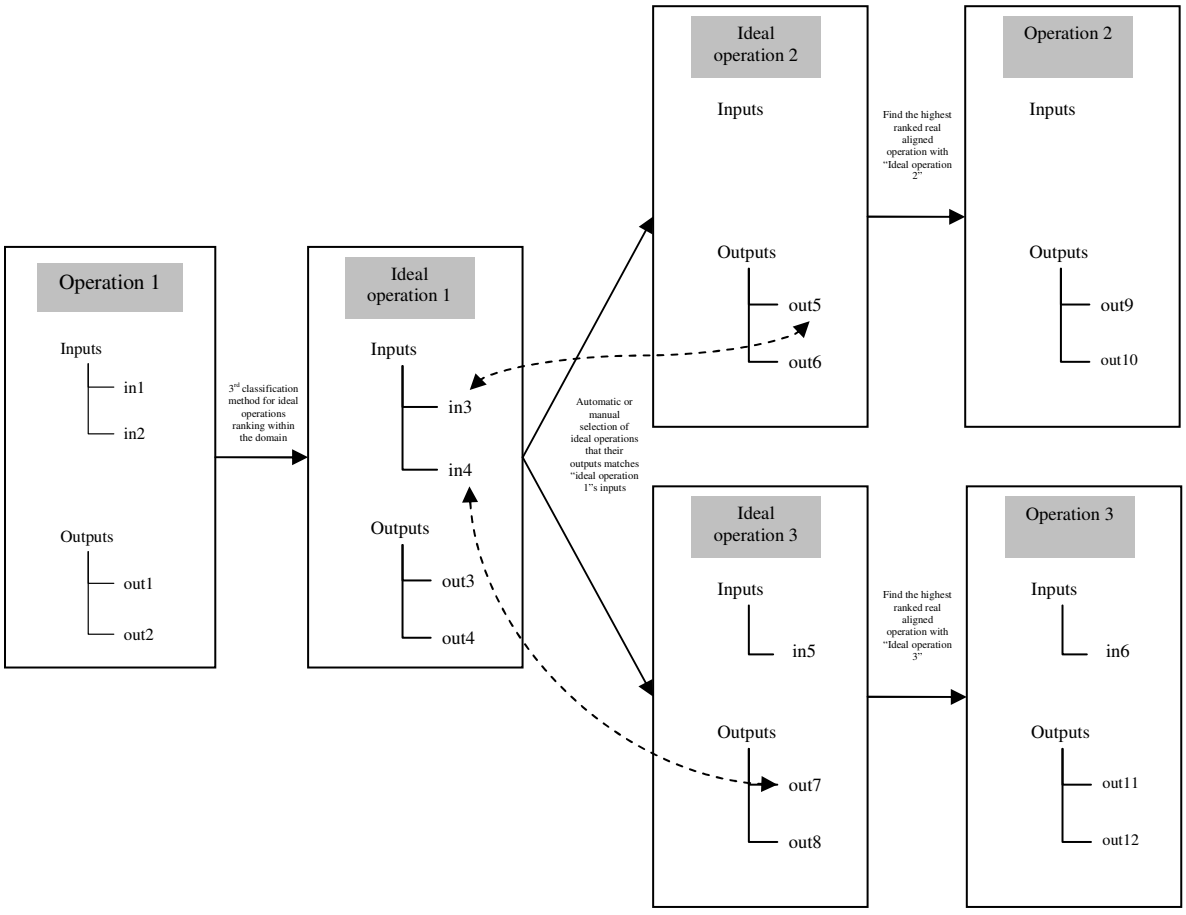


Figure 45: The cascading mechanism using ideal operations matching. The two inputs of the initial operation are finally cascaded with outputs of two already aligned operations in CCM through ideal operation I/O matching. Finally, the inputs needed for "Operation 1" to be invoked is only "in6".

Finally, it should be noted that in the current version of CAAT the linking between the ideal operations is transparent to the users. In a future CAAT version the linking

of input/outputs between ideal operations will be also supported. In this way, any SP will be able to provide web service cascading at the ontology level.

6.2. *Linking Ideal Operations*

CAAT allows the creation of cascades between ideal operations. By selecting the "Link ideal operations" option as it is shown in Figure 5 (option no. 6) the window that is shown in Figure 46 appears. Two lists of all ideal operations that are defined in the ontology are shown in the left and right areas on this window. In order for the user to link the parameter of two ideal operations, which may belong to any available domain, the following restrictions should apply:

- The ideal operations cannot be the same
- Any input parameter can be linked only to an output parameter and vice versa.

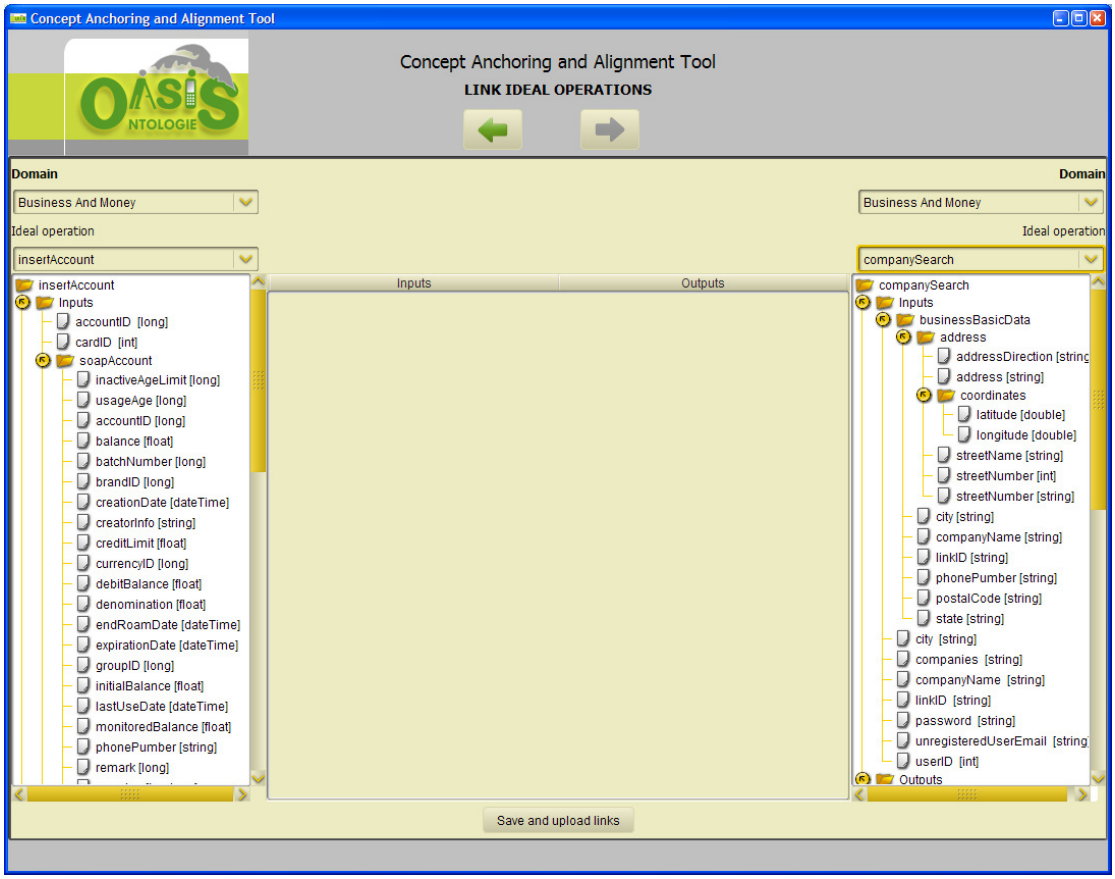


Figure 46: "Link ideal operations" main window.

If any of the above restrictions is not satisfied, error messages such as the one shown in Figure 47 will appear. Linking between two ideal operations is performed dragging from the source (left) ideal operation parameter and dropping to the destination parameter icon (right).

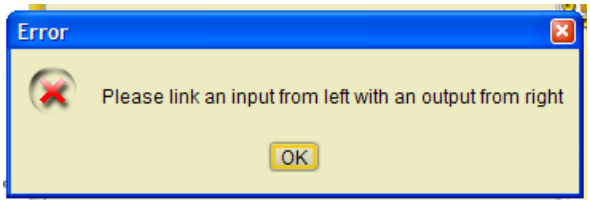


Figure 47: An error message appears when the user tries to align two input or two output parameters.

When the user tries to link two parameters adhering to the aforementioned restrictions, a confirmation window like the one shown in Figure 48 appears. All linked operations through their i/o parameters are shown in the list that appears in the middle area of Figure 49. Once ideal operation linking is complete, the user can save links by pressing the “Save and upload links”. On successful save, the message shown in Figure 50 is shown. The newly created linked operations are stored in the ontology repository.

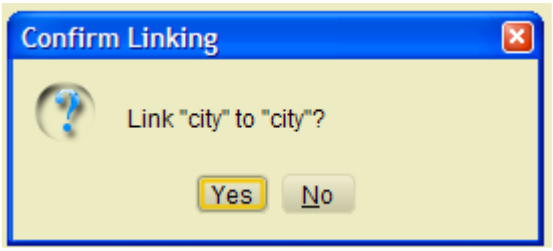


Figure 48: Confirmation window for linking two ideal operation parameters.

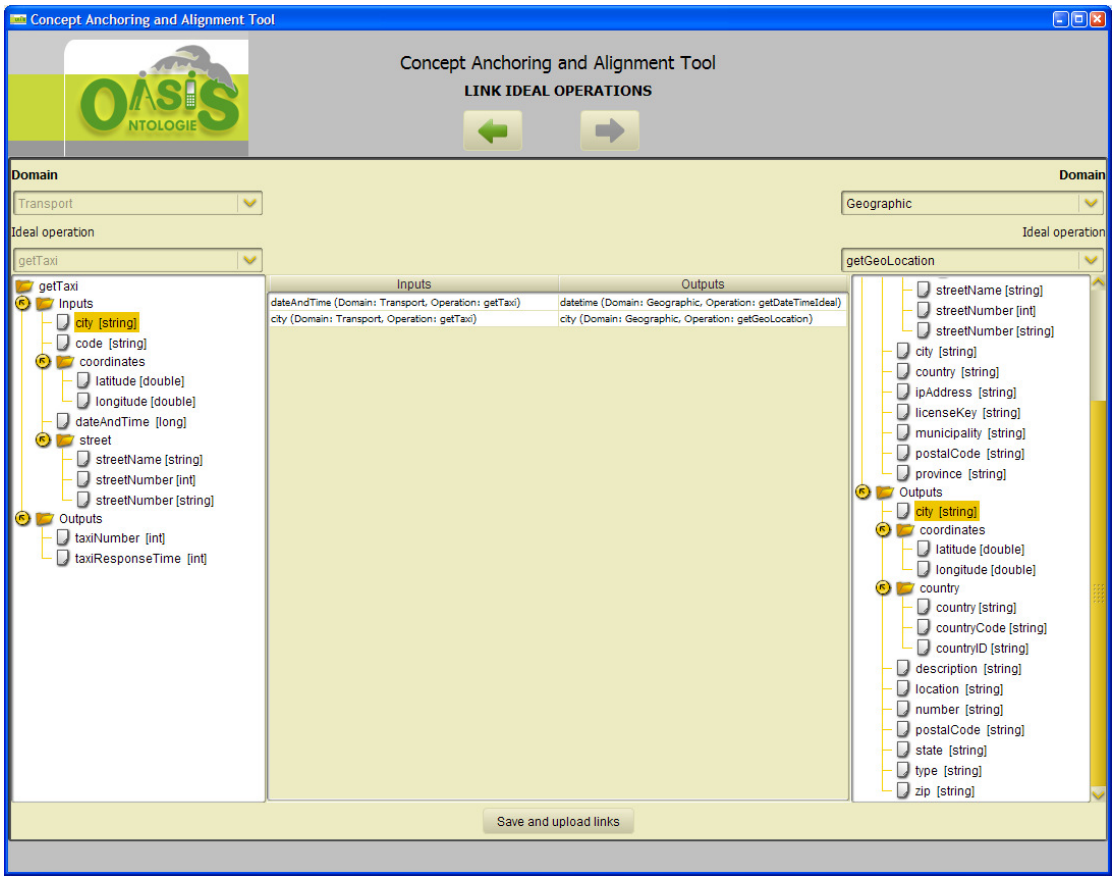


Figure 49: Link ideal operations window.

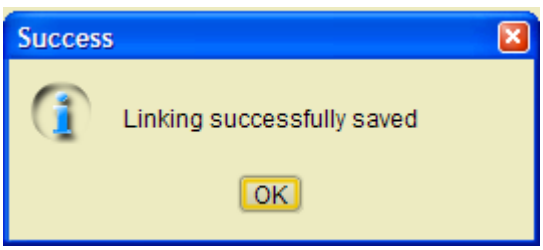


Figure 50: Confirmation window for linking two ideal operation parameters.

6.3. How to Create and Edit a Business Process

In addition to linking ideal operations, which results in the creation of a business process at the ontological level, CAAT provides the ability of cascading real web services and thus creating business processes of multiple web services by different service providers. The result is a composite service with more complex functionalities with as compared to its constituents. The overall web service cascading process is supported in CAAT through its *Business Process Alignment Frame* shown in Figure 51. This can be launched via the "Business processes" option that appears in the main CAAT window.

The canvas area (left) is used for drawing a business process in the form of a workflow, whereas the right area includes a list of toolbars and buttons to support the basic functionality of the business process editor. In summary these are described as follows:

- **Editor mode toolbar**: Three options are supported here:
  - *Edit graph*: this is the edit mode where the user may add new nodes, delete existing ones, etc.
  - *Pick nodes*: it allows moving one node at a time.
  - *Move graph*: it allows moving one or more nodes, as well as the whole diagram.

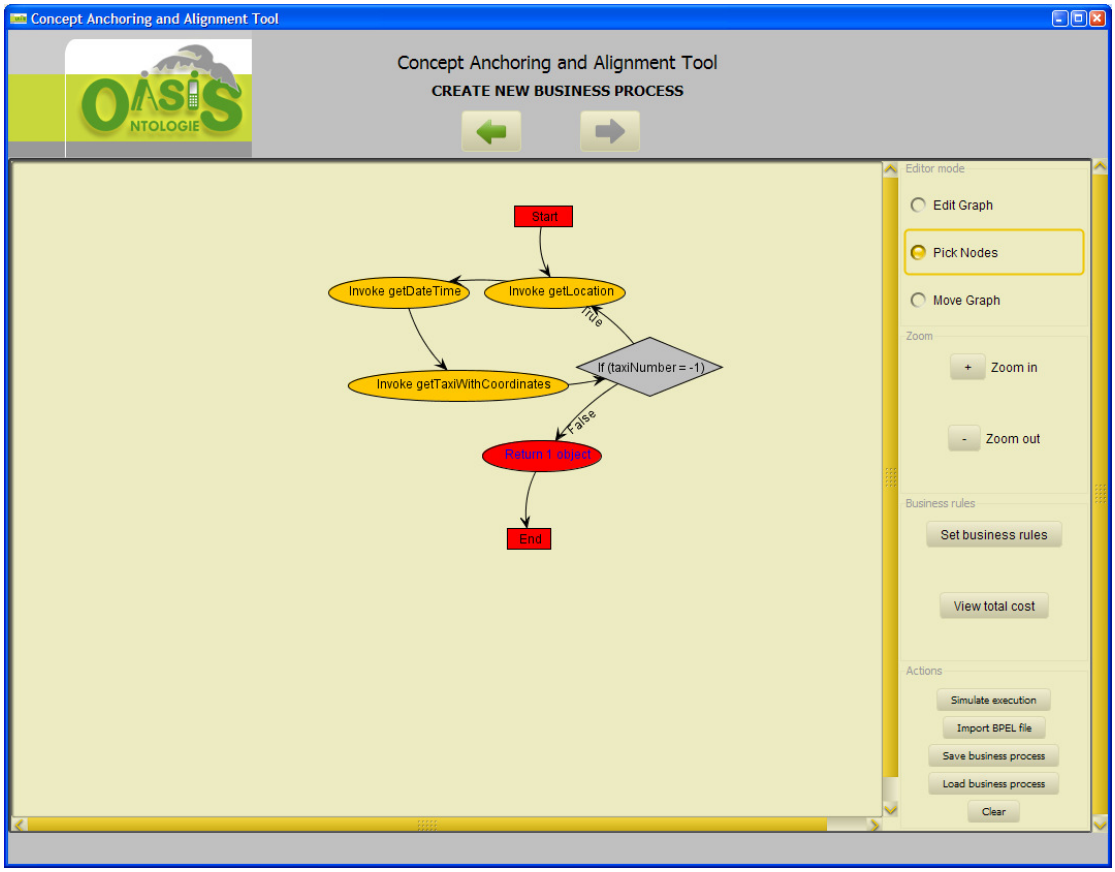


Figure 51: Business process editor. Right toolbox includes the basic operations that the editor supports. In the main area a business process may be designed using basic workflow components.

- **Zoom in/out toolbar**: These buttons allow the user to get a magnified / demagnified view of the business process diagram.

- **Business rules toolbar**: It is used to define business rules (explained in Section 6.4)
- **Actions toolbar**: Includes the following buttons:
  - *Simulate execution*: Executes the current business process
  - *Import BPEL file*: It loads a business process that is saved as a BPEL file.
  - *Load business process*: Loads an existing saved business processes.
  - *Save business process*: Saves the current business process.
  - *Clear*: Cleans up the canvas.

The design of a new business process is performed in the Editor mode toolbar, by using the "Edit Graph" option. Any new business process has a "Start" and an "End" node. A new node can be added by right clicking anywhere on the business process canvas and selecting "Create node". The list of node types that appears is shown in Figure 52. Each node type represents different semantics for the components that constitute a business process. All nodes should be connected by arrows that are drawn automatically in the Edit mode when the user clicks and drags the mouse from one node to another. For each one of them a different colour and shape is used. These are illustrated in Figure 53. The various node types as they appear in a left-to-right order have the following meaning:

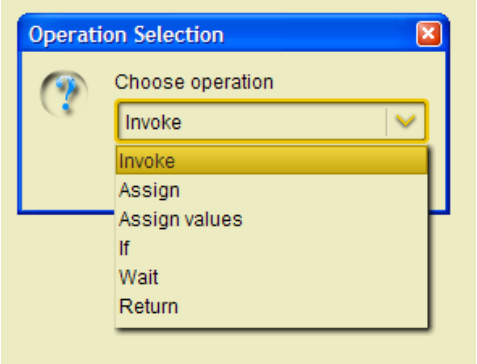


Figure 52: By right clicking on the business process canvas area a list of available options appears.

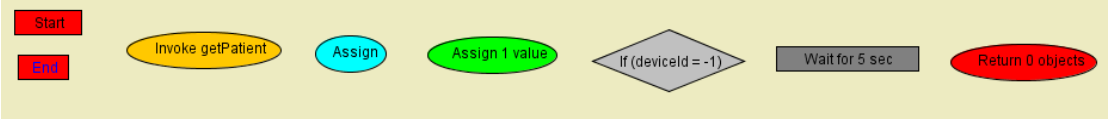


Figure 53: The different types of nodes that may appear in a business process.

- **Start/End**: There is only one Start and only one End node in each business process diagram. The Start node indicates the point at which the business process begins. The End node represents the terminal node of the business process. These nodes appear in any new diagram by default and are obligatory.
- **Invoke**: This type indicates any node that is related to a specific executable operation, defined in a specific web service. When a new Invoke node is added to the diagram, the window shown in Figure 54 appears, that allows the selection of a specific aligned operation to participate in the business process through the particular Invoke node. There are three views. In the first one "By domain" we can see all operations per domain (Figure 54). The second view "By provider" lists all operations per service provider, whereas the third one "Business Processes" shows all previous business processes that have been

saved, so that they can be inserted into a new business process, as well, resulting in more complex business processes.

- **Assign**: This node should be added after at least two Invoke nodes have been previously added in the business process diagram, in order to allow the assignment of input and output parameters with each other. For example, when the output parameter *x* of the *getLocation* operation is assigned to the parameter *latitude* of the *getTaxiWithCoordinates* operation, each time that the business process will be executed, the *getLocation* operation will be invoked before the *getTaxiWithCoordinates* operation, so that the value of its output parameter *x* passes as a value of the input parameter *latitude*. As soon as a new Assign node is inserted in the diagram, a window like the one in Figure 55 appears. Only assignments between input and output parameters are allowed (i.e., no input parameter can be assigned to another input parameters, neither any output parameter is allowed to be assigned to any other output parameter). In order to assign one parameter to another, click on the source (i/o) parameter that appears on the left panel of the window that is shown in Figure 55 and drag it onto the (o/i) destination parameter on the right panel. A confirmation window like the one shown in Figure 56 appears and the i-o mappings appear in the middle panel.

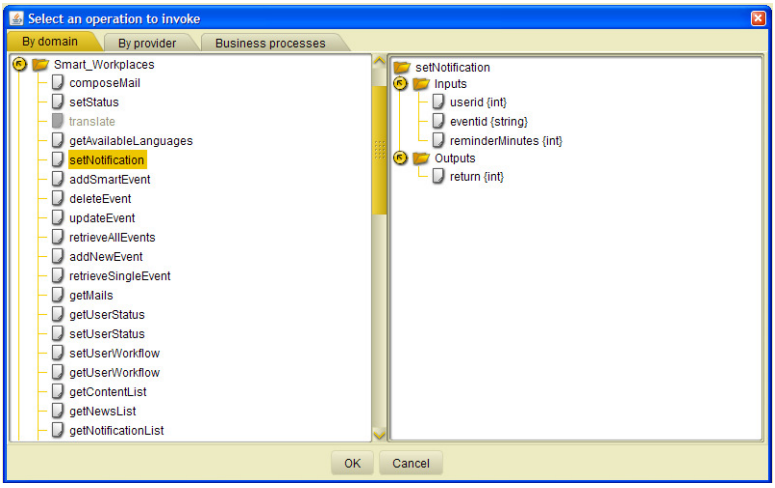


Figure 54: When a new Invoke node is added to the business process a window appears that prompts the user to select, among the aligned operations, the one that should participate in the business process.

- **Assign Values**: This node operates as the Assign node but it allows the assignment of constant values to any input parameter that appears in the business process, e.g. the value '41' is assigned to the input parameter 'age'.
- **If**: This node is used in order to allow conditional execution of a business process. When a new If node is added, the window shown in Figure 57 comes up that allows the creation of a logical expression to be checked during the execution of the business process at the *If* node. E.g. the definition of the *If* statement shown in Figure 57 and Figure 51 implies that when the business process execution workflow meets the *If* node, the value of the *taxiNumber* parameter is checked. If it is found to be equal to -1, which implies that the logical expression "taxiNumber = -1" returns *true* (i.e., that no taxi is available right now), the business process invokes the *getLocation* operation and the execution workflow starts over. Otherwise, the execution flow goes to the

Return node that is used to return an output value by one of the participating operations.

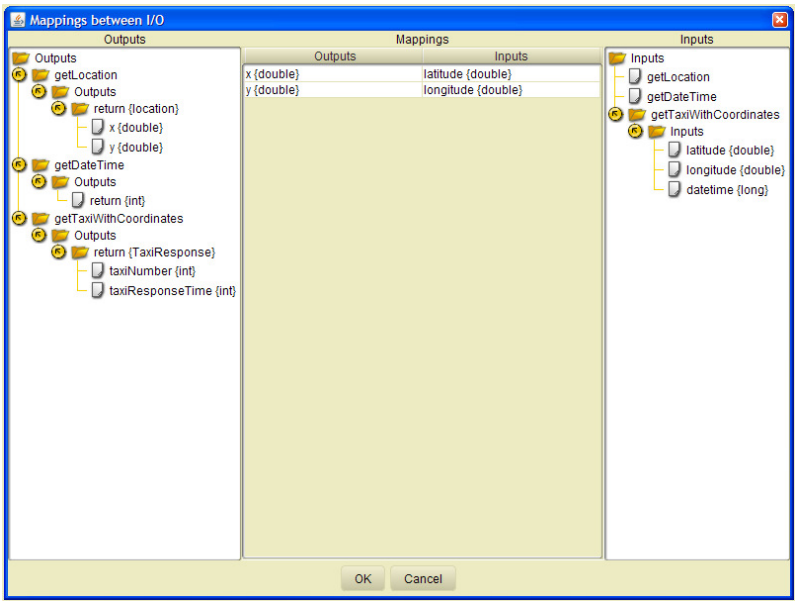


Figure 55: When a new *Invoke* node is added to the business process a prompt window appears.

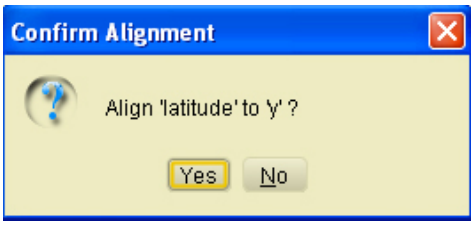


Figure 56: With drag and drop functionality the user can align an input (or an output) shown on the left to an output (or an input, respectively) shown on the right.

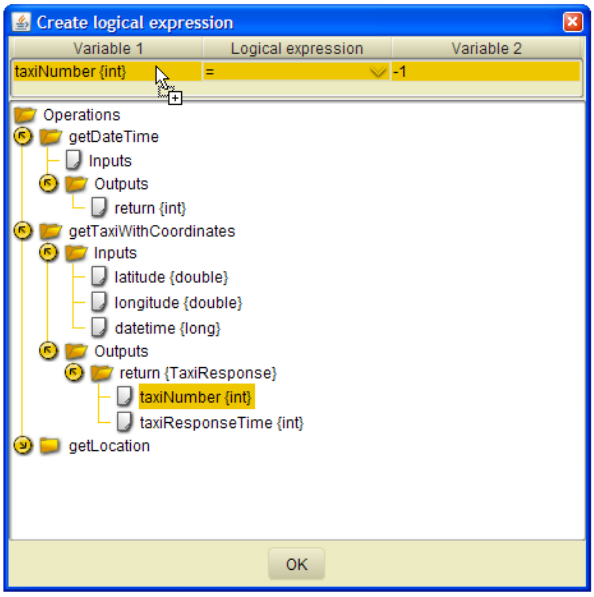


Figure 57: With drag and drop functionality the user can specify an *if* statement (create a logical expression)

- **Wait**: This node is used in order to introduce a delay in the business process. It is quite useful in those cases that one node should wait for some time after the execution of another node, e.g. when a dynamic input is requested as a signal from a GPS sensor in the *getLocation* operation, assuming that the user is moving. The wait node has one parameter that specifies the duration of the delay in sec.
- **Return**: This node is necessary in every business process and should be the last node in a business process. It is used in order to produce as an output a value, which is calculated after the execution of one or more of the involved operations. The user may select which output objects will be returned via a Return node using the window shown in Figure 58 that appears as soon as a new Return node is inserted onto the canvas.

After a business process is created it can be saved by pressing the "Save business process" button located on the Actions toolbar (Figure 59). The GUI prompts the user to enter a name of the business process and select a domain into which the business process should belong. Each time the "Load business process" is pressed, the window that is illustrated in Figure 60 shows a list of all previously saved operations per domain that the user may load and edit.

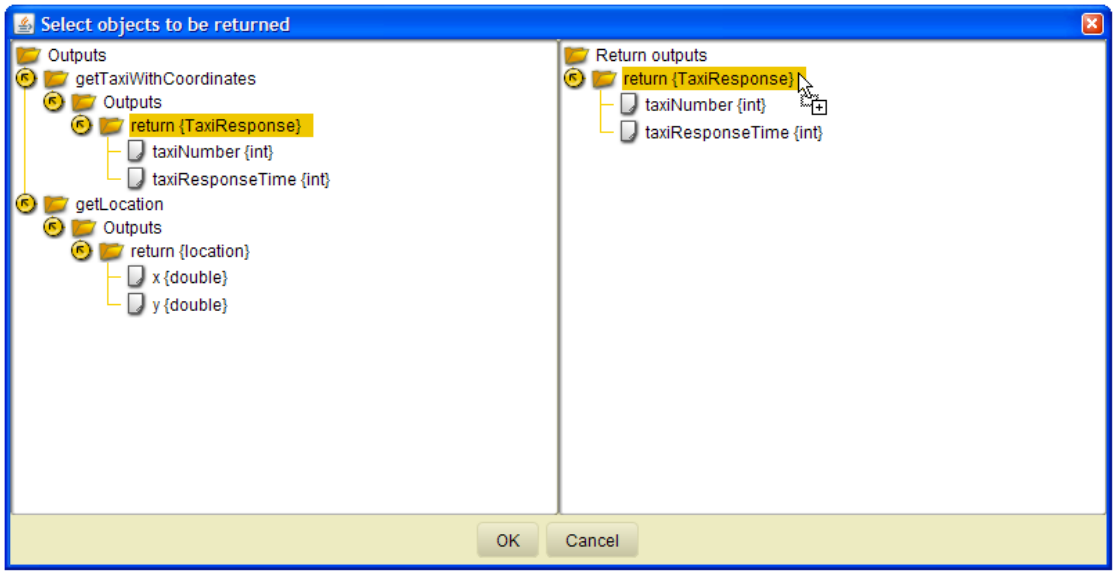


Figure 58: The user may select which output objects will be returned via a *Return* node

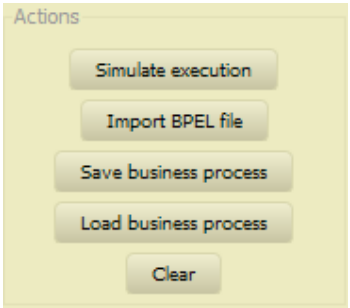


Figure 59: Actions toolbar

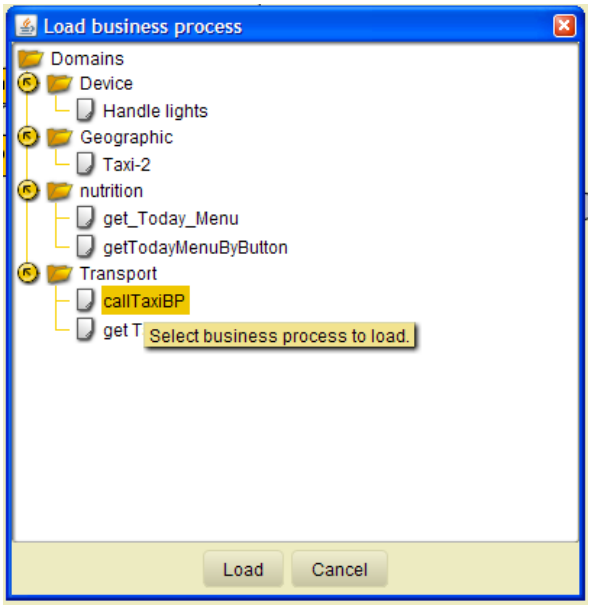


Figure 60: Load business process window

6.4. Business Processes and Business rules

Once a business process is completed and saved, business rules may applied to it in order to determine the pricing policy, define access, etc. on the business process. If business rules have been previously applied on one or more operations that belong to the business process, these operations appear in the Invoke node window (Figure 54) in a different colour.

There are two actions that can be performed with respect to business rules, using the buttons on the Business Rules toolbar shown in Figure 61. The first button "Set business rules" is used to assign rules that determine the pricing policy that corresponds to the use of the overall business process. This is permitted to be done by the service provider who is the owner of the designed business process, on the window shown in Figure 62 that appears by pressing the "Set business rules". This window contains a subset of the options that are available in the Business Rules editor (see Figure 42).

The second button "View total cost" is used to calculate the total cost of a business process, which is equal to the sum of all specific costs at which the separate business process operations may be available, as well as any potential cost assigned to the business process itself. The calculation of total cost can be done at any time, but it is strongly advised to have it done on the completion of the cascading process. By pressing the "View total cost" button on the Business Rules toolbar the window shown Figure 63 appears providing a summary of all costs associated to the business process.

Figure 61: Business rules toolbar

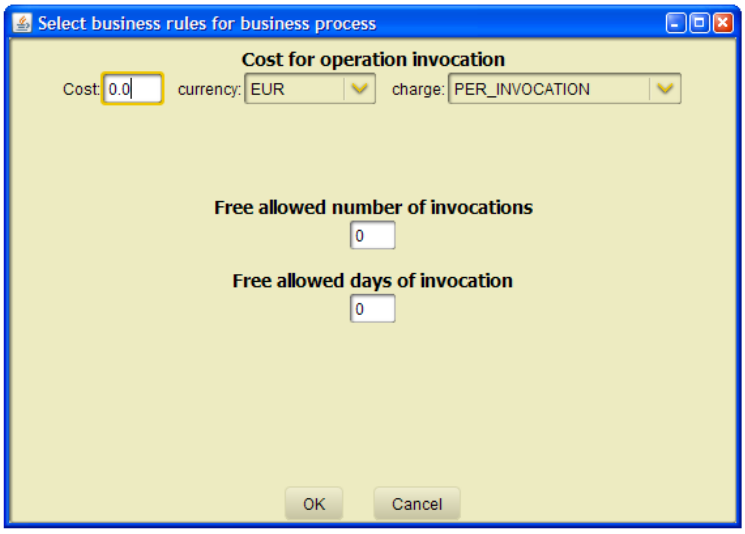


Figure 62: In the "Set business rules" window the pricing policy associated with the use of the business process is determined.

Figure 63: View total cost window.

6.5. *Execution of a business process*

After saving a completely designed business process, we can simulate its execution, i.e. to watch how the business process is executed through the invocation of its subsequent operations. In order to do so, press the "Simulate Execution" button located at the Actions toolbar. In case any of the required inputs, among the involved operations is missing a value (e.g. due to the fact that no appropriate *Assign* node has been defined in the business process) a window such as the one shown in Figure 64 allows manual assignment of input parameter values at runtime. If all input parameters are assigned appropriate values within the business process (through the

corresponding Assign nodes), these will appear in this window as to be disabled (in grey colour). This means that no value is required to be manually assigned to any of the input parameters. In the example shown in Figure 64 the user manually assigns the value "31122012" as an input value into the "datetime" parameter which is of primitive type *long* (i.e. integer number with a high precision).

After pressing the OK button one time, the business process is executed. The execution flow can be viewed on the main canvas window, where each active node in the business process is highlighted in yellow colour, as it is shown in Figure 65. This produces an animated view of the business process execution. In the above example the execution flow is currently passing through the If (response time>5) node, which is highlighted in yellow. Each business process is executed with a default speed slowly enough so that it can be visible by humans. The simulation speed can change using the slide bar located on the Simulation Properties window depicted in Figure 66. This window remains open during the business process execution and closes only as soon as the business process stops. The simulation can be interrupted each time the "Abort simulation" button is pressed. The progress of the execution is also reported on the test area of the Simulation properties window, where a list of log messages appears as the execution goes on.

After the invocation of the business process, the window shown in Figure 67 appears that shows the value of the output (i.e., Return) node of the BP.

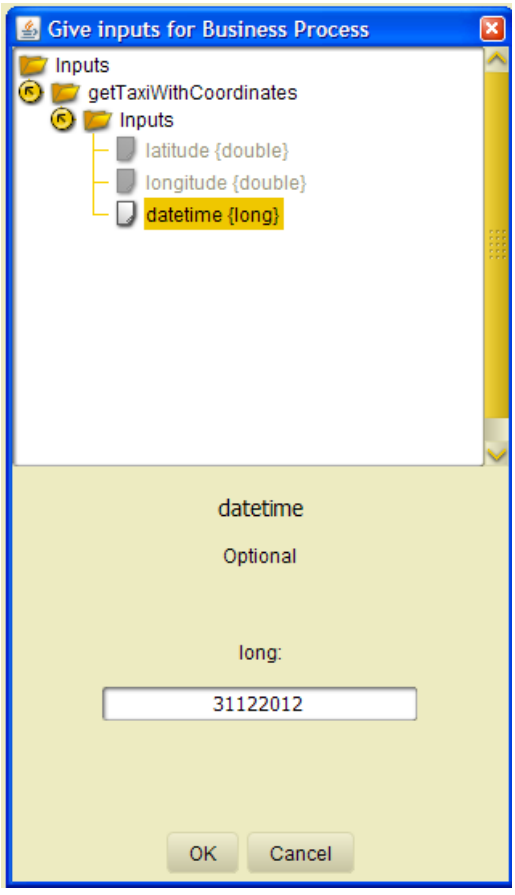


Figure 64: Manual assignment of values to input parameters.

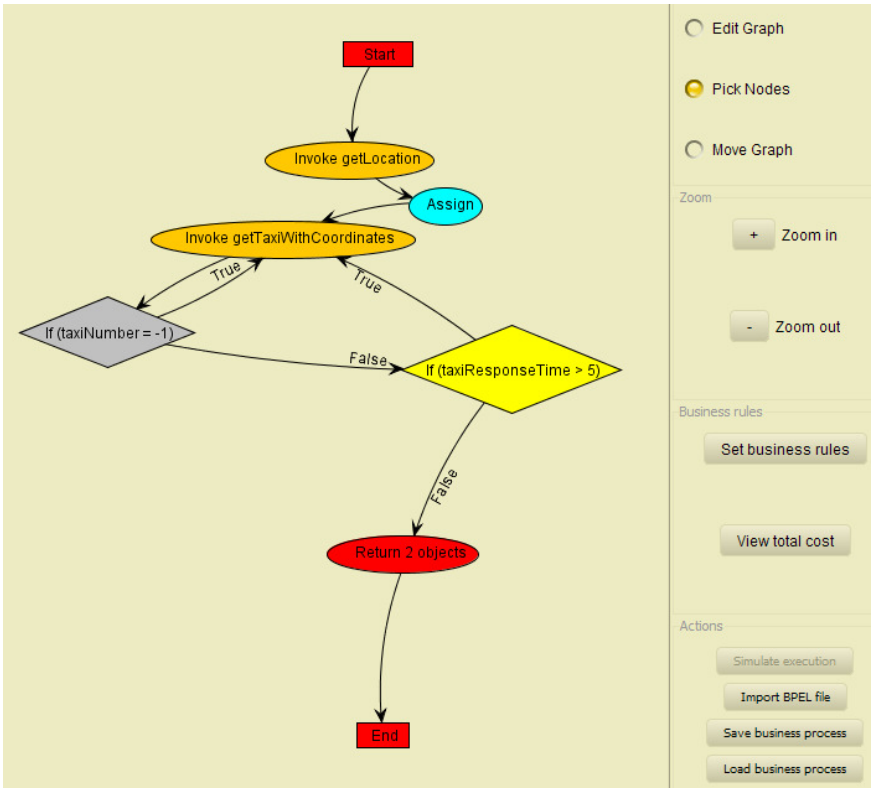


Figure 65: The business process execution is visible on the canvas.

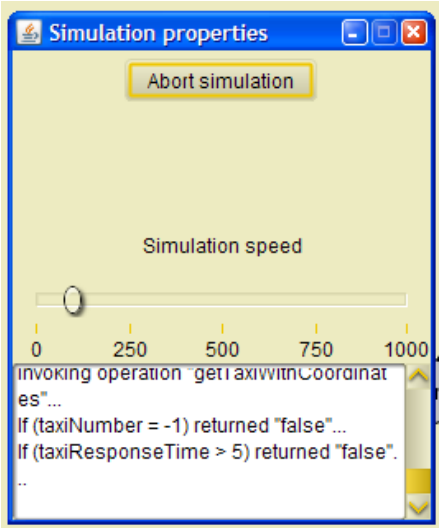


Figure 66: Simulation properties window

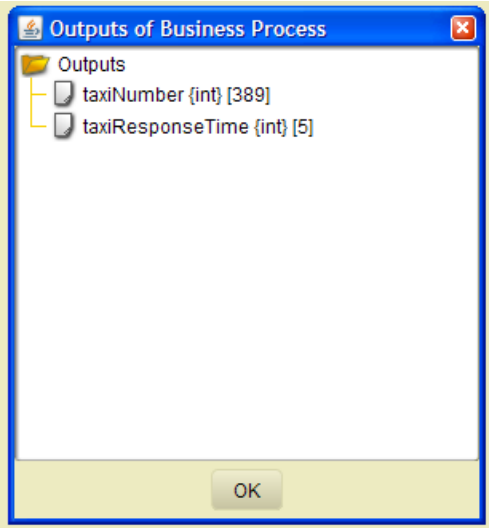


Figure 67: Business process output window

Another way for invoking a business process is via the "*Edit services*" option (no. 4) on the main screen of CAAT (Figure 5). In the example that is shown in Figure 68, the user selects in the *Business Processes* panel (bottom-right) a business process called "*get Taxi from GPS*". By pressing the *Invoke business process* button above, the business process canvas appears (Figure 65) allowing the execution of the business process (through the *Simulate Execution* button).

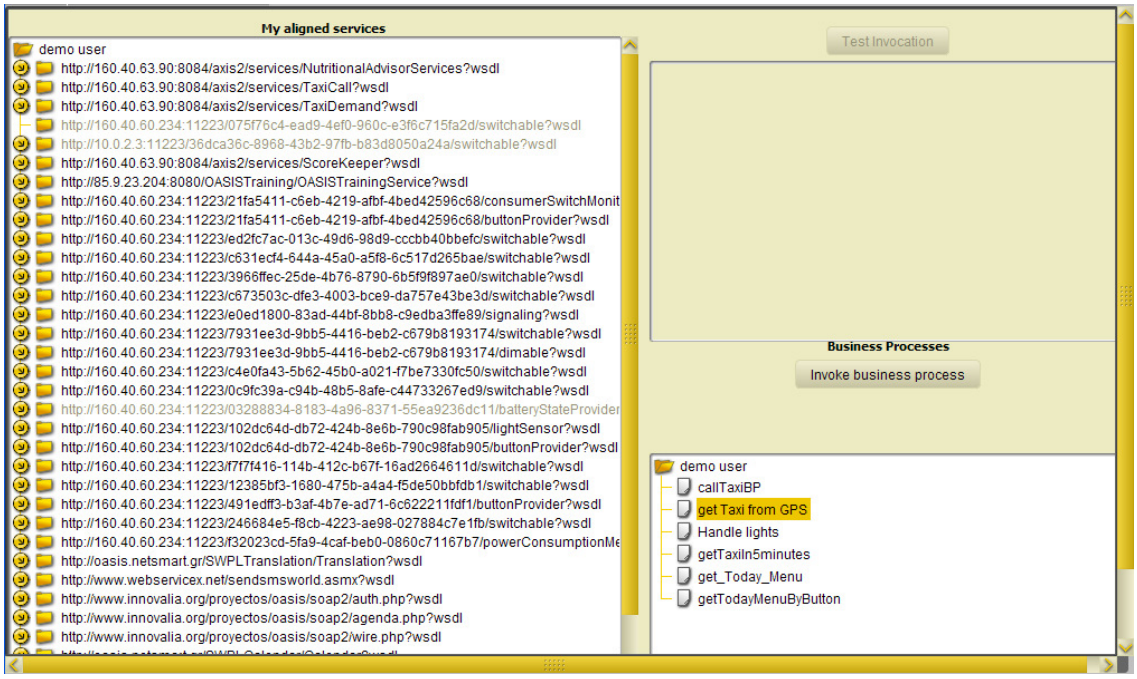


Figure 68: Available business processes of operation “getTaxiWithCoordinates”.

6.6. Importing external BPEL processes

In real life, the business process to be applied may be very complex, i.e. giving different prices to different applied services, depending also upon the frequency of use, time of year, origin of service, type of end-users served, etc; a business rules editor is required to be connected to CCM. For this purpose CAAT supports *BPEL* processes and the ability to import BPEL process that have been created by an external BPEL authoring tool (e.g. *Eclipse BPEL Designer*).

The *Web Services Business Process Execution Language* (WS-BPEL) is a programming language for specifying business properties that involve web services. It is an XML-based language which supports the web services technology stack, including SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination and WS-Transaction. A snapshot of a BPEL process file segment is illustrated in Figure 69.

A BPEL process specifies the exact order in which participating web services should be invoked. This can be done sequentially or in parallel. With BPEL, a conditional behaviour can be expressed, for example, if a web service invocation depends on the value of a previous invocation. It supports also constructing loops, declaring variables, copying and assigning values, defining fault handlers, and so on. By combining all these constructs, new complex business processes can be defined in an algorithmic manner.

By pressing the “Import BPEL” button, which is located on the Actions toolbar (Figure 59) in CAAT business process window, any BPEL process can be imported. As soon as a new BPEL file is loaded in CAAT, the corresponding business process appears as a flowchart on the business process canvas (Figure 51). In this way, any service provider, using any BPEL designer, such as the Netbeans® BPEL Designer depicted in Figure 70, can create a process file, and import it into the CAAT tool and then execute it. CAAT will automatically recognize the sequence of invocations and assignments of the imported process, and integrate as a new Web Service. The

specifications of the BPEL 2.0 language can be found in [2], while general information and examples in [2, 3, 4].

```
<?xml version="1.0" encoding="windows-1253"?>
<process
    name="newProcess"
    targetNamespace="http://enterprise.netbeans.org/bpel/BpelModule2/newProcess"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:sxt="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Trace"
    xmlns:sxed="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor"
    xmlns:tns="http://enterprise.netbeans.org/bpel/BpelModule2/newProcess" xmlns:ns0="http://ws" xmlns:ns1="http://ws/xsd" x
   <import namespace="http://enterprise.netbeans.org/bpel/TaxiRequestWrapper" location="160.40.63.90_8084/axis2/services/Tax
   <import namespace="http://ws" location="160.40.63.90_8084/axis2/services/TaxiRequest.wsdl" importType="http://schemas.xml
   <import namespace="http://enterprise.netbeans.org/bpel/NutritionalAdvisorServicesWrapper" location="160.40.63.90_8084/axi
   <import namespace="http://NutritionalAdvisor" location="160.40.63.90_8084/axis2/services/NutritionalAdvisorServices.wsdl"
   <partnerLinks>
      <partnerLink name="PartnerLink2" xmlns:tns="http://enterprise.netbeans.org/bpel/NutritionalAdvisorServicesWrapper" par
      <partnerLink name="PartnerLink3" xmlns:tns="http://enterprise.netbeans.org/bpel/TaxiRequestWrapper" partnerLinkType="t
      <partnerLink name="PartnerLink1" xmlns:tns="http://enterprise.netbeans.org/bpel/TaxiRequestWrapper" partnerLinkType="t
   </partnerLinks>
   <variables>
      <variable name="GetTaxiWithCoordinatesOut1" messageType="ns0:getTaxiWithCoordinatesResponse"/>
      <variable name="GetTaxiWithCoordinatesIn1" messageType="ns0:getTaxiWithCoordinatesRequest"/>
      <variable name="GetDailyMenuOut" xmlns:ns="http://NutritionalAdvisor" messageType="ns:getDailyMenuResponse"/>
      <variable name="GetDailyMenuIn" xmlns:ns="http://NutritionalAdvisor" messageType="ns:getDailyMenuRequest"/>
      <variable name="GetTaxiWithCoordinatesIn" xmlns:ns="http://ws" messageType="ns:getTaxiWithCoordinatesRequest"/>
      <variable name="GetTaxiWithCoordinatesOut" xmlns:ns="http://ws" messageType="ns:getTaxiWithCoordinatesResponse"/>
   </variables>
   <sequence>
      <receive name="Receive1" createInstance="yes" partnerLink="PartnerLink1" operation="getTaxiWithCoordinates" xmlns:ns="
      <invoke name="Invoke2" partnerLink="PartnerLink3" operation="getTaxiWithCoordinates" portType="ns0:TaxiRequestPortType
      <assign name="Assign1">
         <copy>
            <from>$GetTaxiWithCoordinatesOut1.parameters/ns0:return/ns1:taxiNumber</from>
            <to>$GetDailyMenuIn.parameters/ns2:category</to>
         </copy>
         <copy>
            <from>$GetTaxiWithCoordinatesOut1.parameters/ns0:return/ns1:taxiResponseTime</from>
```

Figure 69: A part of a BPEL process file.

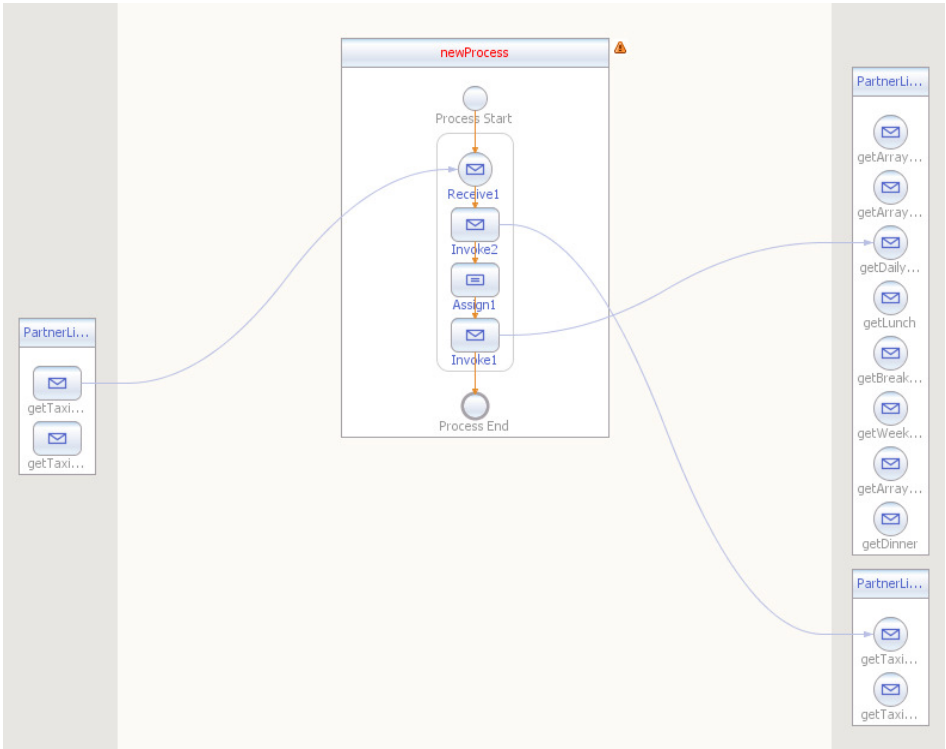


Figure 70: Graphical representation of a BPEL process, in the Netbeans® BPEL designer.

7. References

[1] http://en.wikipedia.org/wiki/Business_process

[2] http://en.wikipedia.org/wiki/Business_Process_Execution_Language

[3] http://docs.REMOTE-open.org/wsbpel/2.0/wsbpel-specification-draft.html

[4] Hewitt E. (2009): Java SOA CookBook, O' Reilly.